Package 'asremlPlus'

May 7, 2026

Version 4.4.58

Date 2026-04-11

Title Augments 'ASReml-R' in Fitting Mixed Models and Packages Generally in Exploring Prediction Differences

Depends R (>= 3.5.0)

Imports dae, devtools, doParallel, dplyr, foreach, ggplot2, graphics, grDevices, methods, nloptr, parallel, qqplotr, RColorBrewer, reshape2, rlang, stats, sticky, stringr, tryCatchLog, utils

Suggests emmeans (>= 1.8.8), lattice, lmerTest, pbkrtest, R.rsp, testthat, tictoc

Enhances asreml

VignetteBuilder R.rsp

SystemRequirements asreml

LazyData true

Description Assists in automating the selection of terms to include in mixed models when 'asreml' is used to fit the models. Procedures are available for choosing models that conform to the hierarchy or marginality principle, for fitting and choosing between two-dimensional spatial models using correlation, natural cubic smoothing spline and P-spline models. A history of the fitting of a sequence of models is kept in a data frame. Also used to compute functions and contrasts of, to investigate differences between and to plot predictions obtained using any model fitting function. The content falls into the following natural groupings: (i) Data, (ii) Model modification functions, (iii) Model selection and description functions, (iv) Model diagnostics and simulation functions, (v) Prediction production and presentation functions, (vi) Response transformation functions, (vii) Object manipulation functions, and (viii) Miscellaneous functions (for further details see 'asremlPlus-package' in help). The 'asreml' package provides a computationally efficient algorithm for fitting a wide range of linear mixed models using Residual Maximum Likelihood. It is a commercial package and a license for it can be purchased from 'VSNi' <https://vsni.co.uk/> as 'asreml-R', who will supply a zip file for local installation/updating (see <https://asreml.kb.vsni.co.uk/>). It is not needed for functions that are methods for 'alldiffs' and 'data.frame' objects. The package 'asremPlus' can also be installed from <http://chris.brien.name/rpackages/>.

License MIT + file LICENSE

URL http://chris.brien.name

BugReports https://github.com/briencj/asremlPlus/issues

NeedsCompilation no

Author Chris Brien [aut, cre] (ORCID: <https://orcid.org/0000-0003-0581-1817>)

Maintainer Chris Brien <chris.brien@adelaide.edu.au>

Repository CRAN

Date/Publication 2026-04-12 16:00:08 UTC

Contents

`asremlPlus-package` *Augments 'ASReml-R' in Fitting Mixed Models and Packages Generally in Exploring Prediction Differences*

Description

Assists in automating the selection of terms to include in mixed models when 'asreml' is used to fit the models. Procedures are available for choosing models that conform to the hierarchy or marginality principle, for fitting and choosing between two-dimensional spatial models using correlation, natural cubic smoothing spline and P-spline models. A history of the fitting of a sequence of models is kept in a data frame. Also used to compute functions and contrasts of, to investigate differences between and to plot predictions obtained using any model fitting function. The content falls into the following natural groupings: (i) Data, (ii) Model modification functions, (iii) Model selection and description functions, (iv) Model diagnostics and simulation functions, (v) Prediction production and presentation functions, (vi) Response transformation functions, (vii) Object manipulation functions, and (viii) Miscellaneous functions (for further details see 'asremlPlus-package' in help). The 'asreml' package provides a computationally efficient algorithm for fitting a wide range of linear mixed models using Residual Maximum Likelihood. It is a commercial package and a license for it can be purchased from 'VSNi' <https://vsni.co.uk/> as 'asreml-R', who will supply a zip file for local installation/updating (see <https://asreml.kb.vsni.co.uk/>). It is not needed for functions that are methods for 'alldiffs' and 'data.frame' objects. The package 'asremPlus' can also be installed from <http://chris.brien.name/rpackages/>.

Version: 4.4.58

Date: 2026-04-11

Index

Note that many of the function below are S3 methods so that the suffix can be omitted. Of course, whether or not the suffix is omitted, the object supplied to the first argument must be of the class specified by the suffix. For example `getFormulae.asreml` is a `getFormulae` method for

an `asreml.object` and so `.asreml` can be omitted and the object supplied to the first argument must be of class `asreml`.

(i) Data

`Oats.dat`

`Wheat.dat`

`WaterRunoff.dat`

(ii) Model modification

`addSpatialModel.asrtests`

`addSpatialModelOnIC.asrtests`

`changeTerms.asrtests`

`iterate.asrtests`

`newfit.asreml`

`reparamSigDevn.asrtests`

`rmboundary.asrtests`

`setvarianceterms.call`

(iii) Model selection and description

`addto.test.summary`
`changeModelOnIC.asrtests`

`chooseModel.asrtests`

`chooseModel.data.frame`

chooseSpatialModelOnIC.asrtests

getTestPvalue.asrtests

infoCriteria.asreml
infoCriteria.list
R2adj.asreml

recalcWaldTab.asrtests

REMLRT.asreml
bootREMLRT.asreml

testranfix.asrtests

testresidual.asrtests

testswapran.asrtests

(iv) Model diagnostics and simulation

plotVariofaces

variofaces.asreml

estimateV.asreml

simulate.asreml

(v) Prediction production and presentation

addBacktransforms.alldiffs

allDifferences.data.frame

exploreLSDs

findLSDminerrors

linTransform.alldiffs

pairdiffsTransform.alldiffs

pickLSDstatistics

plotLSDerrors.data.frame

plotLSDerrors.alldiffs

plotLSDs.data.frame

plotLSDs.alldiffs

plotPredictions.data.frame

plotPvalues.alldiffs

plotPvalues.data.frame
predictPlus.asreml

Forms the predictions for a term, their pairwise differences and associated statistics. A factor having parallel values may occur

ratioTransform.alldiffs

recalcLSD.alldiffs

redoErrorIntervals.alldiffs

renewClassify.alldiffs

sort.alldiffs

subset.alldiffs

sort.predictions.frame

(vi) Response transformation

angular
angular.mod

powerTransform

(vii) Object manipulation

as.alldiffs

asrtests
as.asrtests

as.predictions.frame

convAsremlobj.asreml

convEffectNames2DataFrame.asreml

facCombine.alldiffs

facRecast.alldiffs

facRename.alldiffs

getFormulae.asreml
is.alldiffs

is.asrtests

is.predictions.frame

makeTPPSplineMats.data.frame

print.alldiffs
print.asrtests
print.LSDdata

print.predictions.frame

print.test.summary
print.wald.tab
printFormulae.asreml
sort.alldiffs

subset.alldiffs

subset.list

validAlldiffs
validAsrtests
validPredictionsFrame

(viii) Miscellaneous

getASRemlVersionLoaded

isCompoundSymmetric

loadASRemlVersion
num.recode

permute.square
permute.to.zero.lowertri

The functions whose names end in 'alldiffs" utilize an alldiffs.object that stores: (i) a predictions.frame, being a data frame containing predicted values, variables indexing them and their standard errors and estimability status; the lower and upper limits of error intervals will be included when these are requested, (ii) optionally, square matrices containing all pairwise differences, the standard errors and p-values of the differences, and a data.frame containing LSD values and their summary statistics, (iii) optionally, the variance matrix of the predictions, and (iv) if the response was transformed for analysis, a data frame with backtransforms of the predicted values.

The functions whose names end in 'asrtests', which are most of the model functions, utilize an asrtests.object that stores: (i) the currently fitted model in asreml.obj, (ii) the table of test statistics for the fixed effects in wald.tab, and (iii) a data frame that contains a history of the changes made to the model in test.summary.

Author(s)

Chris Brien [aut, cre] (ORCID: <https://orcid.org/0000-0003-0581-1817>)

Maintainer: Chris Brien <chris.brien@adelaide.edu.au>

References

Butler, D. G., Cullis, B. R., Gilmour, A. R., Gogel, B. J. and Thompson, R. (2023). *ASReml-R Reference Manual Version 4.2.* VSN International Ltd, https://asreml.kb.vsni.co.uk/

See Also

asreml

Examples

```
## Not run:
## Analyse wheat dat using asreml and asremlPlus (see the WheatSpatial Vignette for details)
## Set up for analysis
library(dae)
library(asreml)
library(asremlPlus)
## use ?Wheat.dat for data set details
data(Wheat.dat)

# Add row and column covariates for the spatial modelling
tmp.dat <- within(Wheat.dat,
                  {
                    cColumn <- dae::as.numfac(Column)
                    cColumn <- cColumn  - mean(unique(cColumn))
                    cRow <- dae::as.numfac(Row)
                    cRow <- cRow - mean(unique(cRow))
                  })

# Fit an initial model - Row and column random
current.asr <- do.call(asreml,
                       list(yield ~ Rep + WithinColPairs + Variety,
                            random = ~ Row + Column,
                            residual = ~ Row:Column,
                            data = tmp.dat))

# Intialize a model sequence by loading the current fit into an asrtests object
current.asrt <- as.asrtests(current.asr, NULL, NULL, IClikelihood = "full",
                            label = "Initial model")

# Check for and remove any boundary terms and print a summary of the fit in the asrtests object
current.asrt <- rmboundary(current.asrt)
print(current.asrt)


## Compare a series of information criteria to select a linear mixed model for the data

# Check the need for the term for within Column pairs (a post hoc factor)
current.asrt <- changeModelOnIC(current.asrt, dropFixed = "WithinColPairs",
                           label = "Try dropping withinColPairs", IClikelihood = "full")
print(current.asrt)

# Fit an ar1 model for local spatial variation
spatial.ar1.asrt <- addSpatialModelOnIC(current.asrt, spatial.model = "corr",
                                        row.covar = "cRow", col.covar = "cColumn",
                                        row.factor = "Row", col.factor = "Column",
                                        IClikelihood = "full")
spatial.ar1.asrt <- rmboundary(spatial.ar1.asrt)
infoCriteria(list(nonspatial = current.asrt$asreml.obj,
                  ar1 = spatial.ar1.asrt$asreml.obj))
print(spatial.ar1.asrt)
```

```
# Choose a model for local spatial variation from several potential models
suppressWarnings(
  spatial.asrts <- chooseSpatialModelOnIC(current.asrt,
                                          row.covar = "cRow", col.covar = "cColumn",
                                          row.factor = "Row", col.factor = "Column",
                                          dropRandom = "Row + Column",
                                          rotateX = TRUE, ngridangles = NULL,
                                          asreml.option = "grp", return.asrts = "all"))
# Output the results
print(spatial.asrts$spatial.IC)
print(R2adj(spatial.asrts$asrts$TPNCSS$asreml.obj, include.which.random = ~ .))
print(spatial.asrts$best.spatial.mod)
print(spatial.asrts$asrts$TPNCSS)
printFormulae(spatial.asrts$asrts$TPNCSS$asreml.obj)

## Diagnosting checking using residual plots and variofaces

# Get current fitted asreml object and update to include standardized residuals
current.asr <- spatial.asrts$asrts$TPNCSS$asreml.obj
current.asr <- update(current.asr, aom=TRUE)
Wheat.dat$res <- residuals(current.asr, type = "stdCond")
Wheat.dat$fit <- fitted(current.asr)

# Do residuals-versus-fitted values plot
with(Wheat.dat, plot(fit, res))

# Plot variofaces
variofaces(current.asr, V=NULL, units="addtores",
           maxiter=50, update = FALSE,
           ncores = parallel::detectCores())

# Plot normal quantile plot
ggplot(data = Wheat.dat, mapping = aes(sample = res)) +
  stat_qq_band(bandType = "ts") + stat_qq_line() + stat_qq_point() +
  labs(x = "Theoretical Quantiles", y = "Sample Quantiles",
       title = "Normal probability plot") +
  theme(plot.title = element_text(size = 12, face = "bold")) + theme_bw()

## Prediction production and presentation

# Get Variety predictions and all pairwise prediction differences and p-values
Var.diffs <- predictPlus(classify = "Variety",
                         asreml.obj=current.asr,
                         error.intervals="halfLeast",
                         wald.tab=current.asrt$wald.tab,
                         sortFactor = "Variety",
                         tables = "predictions")

# Plot the Variety predictions, with halfLSD intervals, and the p-values
plotPredictions(Var.diffs$predictions,
                classify = "Variety", y = "predicted.value",
                error.intervals = "half")
plotPvalues(Var.diffs)
```

```
## End(Not run)
```

`addBacktransforms.alldiffs` *Adds or recalculates the backtransforms component of an* `alldiffs.object`.

Description

Given an `alldiffs.object`, adds or recalculate its `backtransforms` component. The values of `transform.power`, `offset`, `scale` and `transform.function` from the `backtransforms` component will be used, unless this component is `NULL` when the values supplied in the call will be used.

Usage

```
## S3 method for class 'alldiffs'
addBacktransforms(alldiffs.obj,
                  transform.power = 1, offset = 0, scale = 1,
                  transform.function =  "identity", ...)
```

Arguments

`alldiffs.obj`	An `alldiffs.object`.
`transform.power`	A `numeric` specifying the power of a transformation, if one has been applied to the response variable. Unless it is equal to 1, the default, back-transforms of the predictions will be obtained and presented in tables or graphs as appropriate. The back-transformation raises the predictions to the power equal to the reciprocal of `transform.power`, unless it equals 0 in which case the exponential of the predictions is taken.
`offset`	A `numeric` that has been added to each value of the response after any scaling and before applying any power transformation.
`scale`	A `numeric` by which each value of the response has been multiplied before adding any offset and applying any power transformation.
`transform.function`	A `character` giving the name of a function that specifies the scale on which the predicted values are defined. This may be the result of a transformation of the data using the function or the use of the function as a link function in the fitting of a generalized linear (mixed) model (GL(M)M). The possible `transform.functions` are `identity`, `log`, `inverse`, `sqrt`, `logit`, `probit`, and `cloglog`. The `predicted.values` and `error.intervals`, if not `StandardError` intervals, will be back-transformed using the inverse function of the `transform.function`. The `standard.error` column will be set to `NA`, unless (i) `asreml` returns columns named `transformed.value` and `approx.se`, as well as those called `predicted.values` and `standard.error` (such as when a GLM is fitted) and (ii) the values in

transformed.value are equal to those obtained by backtransforming the predicted.values using the inverse function of the transform.function. Then, the approx.se values will be saved in the standard.error column of the backtransforms component of the returned alldiffs.obj. Also, the transformed.value and approx.se columns are removed from both the predictions and backtransforms components of the alldiffs.obj. Note that the values that end up in the standard errors column are approximate for the backtransformed values and are not used in calculating error.intervals.

... Provision for passing arguments to functions called internally - not used at present.

Value

An alldiffs.object with components predictions, vcov, differences, p.differences, sed, LSD and backtransforms.

The backtransforms component will have the attributes (i) LSDtype, LSDby and LSDstatistic added from the predictions component and (ii) transform.power, offset, scale, and link.

Author(s)

Chris Brien

See Also

asremlPlus-package, as.alldiffs, sort.alldiffs, subset.alldiffs, print.alldiffs, renewClassify.alldiffs, redoErrorIntervals.alldiffs, plotPredictions.data.frame, predictPlus.asreml, predictPresent.asreml

Examples

```
##Subset WaterRunoff data to reduce time to execute
data(WaterRunoff.dat)
tmp <- subset(WaterRunoff.dat, Date == "05-18" & Benches != "3")

##Use asreml to get predictions and associated statistics

## Not run:
asreml.options(keep.order = TRUE) #required for asreml-R4 only
current.asr <- asreml(fixed = log.Turbidity ~ Benches + (Sources * (Type + Species)),
                      random = ~ Benches:MainPlots,
                      keep.order=TRUE, data= tmp)
current.asrt <- as.asrtests(current.asr, NULL, NULL)
TS.diffs <- predictPlus(classify = "Sources:Type",
                        asreml.obj = current.asr,
                        wald.tab = current.asrt$wald.tab,
                        present = c("Sources", "Type", "Species"))

## End(Not run)

##Use lmeTest and emmmeans to get predictions and associated statistics
```

```
if (requireNamespace("lmerTest", quietly = TRUE) &&
    requireNamespace("emmeans", quietly = TRUE))
{
  m1.lmer <- lmerTest::lmer(log.Turbidity ~ Benches + (Sources * (Type + Species)) +
                              (1|Benches:MainPlots),
                            data=tmp)
  TS.emm <- emmeans::emmeans(m1.lmer, specs = ~ Sources:Species)
  TS.preds <- summary(TS.emm)
  den.df <- min(TS.preds$df, na.rm = TRUE)
  ## Modify TS.preds to be compatible with a predictions.frame
  TS.preds <- as.predictions.frame(TS.preds, predictions = "emmean",
                                   se = "SE", interval.type = "CI",
                                   interval.names = c("lower.CL", "upper.CL"))

  ## Form an all.diffs object and check its validity
  TS.vcov <- vcov(TS.emm)
  TS.diffs <- allDifferences(predictions = TS.preds, classify = "Sources:Species",
                             vcov = TS.vcov, tdf = den.df)
  validAlldiffs(TS.diffs)
}

## Recalculate the back-transforms of the predictions obtained using asreml or lmerTest
if (exists("TS.diffs"))
{
  TS.diffs <- addBacktransforms.alldiffs(TS.diffs, transform.power = 0)
}
```

`addSpatialModel.asrtests` *Adds, to a supplied model, a spatial model that accounts for local spatial variation.*

Description

Adds either a correlation, two-dimensional tensor-product natural cubic smoothing spline (TP-NCSS), or a two-dimensional tensor-product penalized P-spline model (TPPS) to account for the local spatial variation exhibited by a response variable measured on a potentially irregular grid of rows and columns of the units. The data may be arranged in sections, for each of which there is a grid and for which the model is to be fitted separately. Also, the rows and columns of a grid are not necessarily one observational unit wide. For `TPPS` models for which the order of differencing the penalty matrix is two, the an optimal rotation of the null-space eigenvectors of the penalty matrix can be investigated.

No hypothesis testing or comparison of information criteria is made. To use information criteria to decide whether to change the model use `chooseSpatialModelOnIC.asrtests`.

The model fit supplied in the `asrtests.obj` should not include terms that will be included in the local spatial model. All spatial model terms are fitted as fixed or random. Consequently, the residual model does not have to be iid.

One or more rows is added for each section to the test.summary data.frame. Convergence and the occurrence of fixed correlations in fitting the model is checked and a note included in the action if there was not. All components of the asrtests.object are updated for the new model.

Usage

```
## S3 method for class 'asrtests'
addSpatialModel(asrtests.obj, spatial.model = "TPPS",
                sections = NULL,
                row.covar = "cRow", col.covar = "cCol",
                row.factor = "Row", col.factor = "Col",
                corr.funcs = c("ar1", "ar1"), corr.orders = c(0, 0),
                row.corrFitfirst = TRUE,
                allow.corrsJointFit = TRUE, nugget.variance = TRUE,
                dropFixed = NULL, dropRandom = NULL,
                nsegs = NULL, nestorder = c(1,1),
                degree = c(3,3), differorder = c(2,2),
                usRandLinCoeffs = TRUE,
                rotateX = FALSE, ngridangles = NULL,
                which.rotacriterion = "AIC", nrotacores = 1,
                asreml.option = "grp", tpps4mbf.obj = NULL,
                allow.unconverged = TRUE, allow.fixedcorrelation = TRUE,
                checkboundaryonly = FALSE, update = TRUE, trace = FALSE,
                maxit = 30, IClikelihood = "full", which.IC = "AIC", ...)
```

Arguments

asrtests.obj	An asrtests.object containing the components (i) asreml.obj, (ii) wald.tab, and (iii) test.summary.
spatial.model	A single character string nominating the type of spatial model to fit. Possible values are corr, TPNCSS and TPPS.
sections	A single character string that specifies the name of the column in the data.frame that contains the factor that identifies different sections of the data to which separate spatial models are to be fitted. Note that, for other terms that involve sections in the random formula, there should be separate terms for each level of sections. For example, in a blocked experiment involving multiple sites, there should be the sum of separate terms for the Blocks at each Site i.e. a formula that contains terms like at(Site, i):Block for each site and these are separated by '+'. Otherwise, the combined term (e.g. Site:Block) will impact on the fitting of the local spatial models for the different Sites. Similarly, a separate residual variance for each of the sections should be fitted, unless there is a need to fit a different variance structure to the residual, e.g. heterogeneous residual variances depending on treatments. Separate residual variances for sections can be achieved using the asreml functions dsum or idh. Because, unlike random terms, terms for residual variances are not removed from the model, compound residual terms can be used to include them in the model, e.g. terms with idh or dsum with multiple levels in the list or leaving levels out altogether. In addition to allowing the independent fitting of models to the sections, sepa-

rate residual variance terms allows a nugget variance to be fitted in a correlation model for each of the `sections`.

`row.covar` A single `character` string nominating a `numeric` that contains the values of a centred covariate indexing the rows of a grid. The `numeric` must be a column in the `data.frame` stored in the `asreml.obj` that is a component of the supplied `asrtests.obj`.

`col.covar` A single `character` string nominating a `numeric` that contains the values of a centred covariate indexing the columns of a grid. The `numeric` must be a column in the `data.frame` stored in the `asreml.obj` that is a component of the supplied `asrtests.obj`.

`row.factor` A single `character` string nominating a `factor` that indexes the rows of a grid that are to be one dimension of a spatial correlation model. The `factor` must a column in the `data.frame` stored in the `asreml.obj` that is a component of the supplied `asrtests.obj`.

`col.factor` A single `character` string nominating a `factor` that indexes the columns of a grid that are to be one dimension of a spatial correlation model. The `factor` must a column in the `data.frame` stored in the `asreml.obj` that is a component of the supplied `asrtests.obj`.

`corr.funcs` A single `character` string of length two that specifies the `asreml` one-dimensional correlation or variance model function for the row and column dimensions of a two-dimensional separable spatial correlation model to be fitted when `spatial.model` is `corr`; the two-dimensional model is fitted as a random term. If a correlation or variance model is not to be investigated for one of the dimensions, specify `""` for that dimension. If the correlation model is `corb`, the values of `corr.orders` are used for its order argument (`b`).

`corr.orders` A `numeric` of length two that specifies the order argument (`b`) values for the row and column dimensions of a two-dimensional separable spatial correlation model when `spatial.model` is `corr` and the `corr.funcs` for a dimension is `corb`, the `asreml` banded correlation model. If one of the dimensions does not involve an order argument, set the value of `corr.orders` for that dimension to zero. For a dimension for which the `corr.funcs` is `corb` and `corr.orders` is zero, a model with a single band, the correlation between immediate neighbours, will be fitted and then further bands, up to a maximum of 10 bands, will be added until the addition of an extra band does not reduce the information criterion nominated using `which.IC`. Note that the two-dimensional spatial model is fitted as a random term.

`row.corrFitfirst`

A `logical`. If `TRUE` then, in fitting the model for `spatial.model` set to `corr`, the row correlation or variance function is fitted first, followed by the addition of the column correlation or variance function. If `FALSE`, the order of fitting is reversed.

`allow.corrsJointFit`

A `logical` which, if `TRUE`, will allow the simultaneous fitting of correlation functions for the two dimensions of the grid when separate fits have failed to fit any correlation functions. This argument is available for when a joint fit hangs the system.

nugget.variance

A logical which, if TRUE, will result in an attempt to fit a nugget or unit-specific variance. Otherwise, a nugget or unit-specific variance will not be fitted.

dropFixed

A single character string or a character vector of strings with an element for each level of sections in the same order as the sections levels. Each string, which if it is not NA and after the addition of ". ~ . -" and conversion to a formula that is then expanded, specifies the sum of a set of terms to be dropped from the fixed formula in fitting splines (TPPS and TPNCSS). The result is that the fitted model supplied in the asrtests.obj, that includes these terms, will be compared with the fitted model that has had them removed and a spatial model added.

An element that is NA indicates that no term pertaining to the corresponding sections level is to be removed. If sections is not NULL and a single character string has been supplied, the terms specified by the string are taken to be terms that are independent of the sections and will be removed when adding the spatial model for the first sections.

The terms must match those in the wald.tab component of the asrtests.obj. The fixed terms will be reordered so that single-variable terms come first, followed by two-variable terms and so on. Note also that multiple terms specified using a single asreml::at function can only be dropped as a whole. If the term was specified using an asreml::at function with a single level, then it can be removed and either the level itself or its numeric position in the levels returned by the levels function can be specified.

dropRandom

A single character string or a character vector of strings with an element for each level of sections in the same order as the sections levels. Each string, which if it is not NA and after the addition of " ~ . -" and conversion to a formula that is then expanded, specifies the sum of a set of terms to be dropped from the random formula in fitting splines (TPPS and TPNCSS). The result is that the fitted model supplied in the asrtests.obj, that includes these terms, will be compared with the fitted model that has had them removed and a spatial model added.

An element that is NA indicates that no term pertaining to the corresponding sections level is to be removed. If sections is not NULL and a single character string has been supplied, the terms specified by the string are taken to be terms that are independent of the sections and will be removed when adding the spatial model for the first sections.

The terms must match those in the vparameters component of the asreml.obj component in the asrtests.obj. Note also that multiple terms specified using a single asreml::at function can only be dropped as a whole. If the term was specified using an asreml::at function with a single level, then it can be removed and either the level itself or its numeric position in the levels returned by the levels function can be specified.

nsegs

A pair of numeric values giving the number of segments into which the column and row ranges are to be split, respectively, for fitting a P-spline model (TPPS) (each value specifies the number of internal knots + 1). If not specified, then (number of unique values - 1) is used in each dimension; for a grid layout with equal spacing, this gives a knot at each data value. If sections is not NULL

and the grid differs between the sections, then nsegs will differ between the sections.

nestorder A numeric of length 2. The order of nesting for column and row dimensions, respectively, in fitting a P-spline model (TPPS). A value of 1 specifies no nesting, a value of 2 generates a spline with half the number of segments in that dimension, etc. The number of segments in each direction must be a multiple of the order of nesting.

degree A numeric of length 2. The degree of polynomial spline to be used for column and row dimensions respectively, in fitting a P-spline (TPPS).

difforder A numeric of length 2. The order of differencing for column and row dimensions, respectively, in fitting a P-spline (TPPS).

usRandLinCoeffs

A logical which, if TRUE and a P-spline (TPPS) is being fittted, will attempt to fit an unstructured variance model to the constant and linear terms in the interactions for constant and linear terms in one grid dimension interacting with smooth terms in the second grid dimension. The unstructured variance model can only be fitted if both the constant and linear interaction terms have been retained in the fitted model. This argument can be used to omit the attempt to fit an unstructured variance model when the attempt results in a system error.

rotateX A logical indicating whether to rotate the eigenvectors of the penalty matrix, as described by Piepho, Boer and Williams (2022), when fitting a P-spline (TPPS). Setting rotateX to TRUE results in a search for an optimized rotation under a model that omits the random spline interaction terms. If ngridangles is set to NULL, the optimal rotation us found using an optimizer (nloptr::bobyqa). Otherwise, the optimal rotation is found by exploring the fit over a two-dimensional grid of rotation angle pairs. The optimization seeks to optimize the criterion nominated in which.rotacriterion. Rotation of the eigenvectors is only relevant for difforder values greater than 1 and has only been implemented for difforder equal to 2.

ngridangles A numeric of length 2. If NULL (the default), the optimal pair of angles for rotating the eigenvectors of the penalty matrix of a P-spline (TPPS) will be determined using a nonlinear optimizer (nloptr::bobyqa). Otherwise, its two values specify the numbers of angles between 0 and 90 degrees for each of the row and column dimensions to be used in determining the optimal pair of angles. Specifying factors of 90 will result in integer-valued angles. The number of grid points, and hence re-analyses will be the product of the values of (ngridangles + 1).

which.rotacriterion

A single character string nominating which of the criteria, out of the deviance, the likelihood, the AIC and the BIC, is to be used in determining the optimal rotation of the eigenvectors of the penalty matrix. The deviance uses the REML value computed by asreml; the other criteria use the full likelihood, evaluated using the REML estimates, that is computed by infoCriteria.asreml.

nrotacores A numeric specifying the number of cores to deploy for running the analyses required to search the two-dimensional grid of rotation angles when rotateX is TRUE. Parallel processing has been implemented for analyzing, for each column

angle, the set of angles to be investigated for the row dimension. The default value of one means that parallel processing will not be used. The value chosen for nrotacores needs to balanced against the other processes that are using parallel processing at the same time.

asreml.option A single character string specifying whether the grp or mbf methods are to be used to supply externally formed covariate matrices to asreml when fitting a P-spline (TPPS). Compared to the mbf method, the grp method is somewhat faster, but creates large asrtests.objects for which the time it takes to save them can exceed any gains in execution speed. The grp method adds columns to the data.frame containing the data. On the other hand, the mbf method adds only the fixed covariates to data and stores the random covariates in the environment of the internal function that calls the spline-fitting function; there are three smaller data.frames for each section that are not stored in the asreml.object resulting from the fitted model.

tpps4mbf.obj An object made with makeTPPSplineMats.data.frame that contains the spline basis information for fitting P-splines. The argument tpps4mbf.obj only needs to be set when the mbf option of asreml.option is being used and it is desired to use mbf data.frames that have been created and stored prior to calling addSpatialModel.asrtests. If tpps4mbf.obj is NULL, makeTPPSplineMats.data.frame will be called internally to produce the required mbf data.frames.

allow.unconverged
A logical indicating whether to accept a new model even when it does not converge. If FALSE and the fit of the new model does not converge, the supplied asrtests.obj is returned. Also, if FALSE and the fit of the new model has converged, but that of the old model has not, the new model will be accepted.

allow.fixedcorrelation
A logical indicating whether to accept a new model even when it contains correlations in the model whose values have been designated as fixed, bound or singular. If FALSE and the new model contains correlations whose values have not been able to be estimated, the supplied asrtests.obj is returned. The fit in the asreml.obj component of the supplied asrtests.obj will also be tested and a warning issued if both fixed correlations are found in it and allow.fixedcorrelation is FALSE.

checkboundaryonly
If TRUE then boundary and singular terms are not removed by rmboundary.asrtests; a warning is issued instead. Note that, for correlation models, the fitting of each dimension and the test for a nugget term are performed with checkboundaryonly set to TRUE and its supplied setting only honoured using a call to rmboundary.asrtests immediately prior to returning the final result of the fitting.

update If TRUE, then newfit.asreml is called to fit the model to be tested, using the values of the variance parameters stored in the asreml.object, that is stored in asrtests.obj, as starting values. If FALSE, then newfit.asreml will not use the stored variance parameter values as starting values when fitting the new model, the only modifications being (i) to add the terms for the spatial models and (ii) those specified via ....

trace If TRUE then the stages in fitting a correlation model are displayed.

maxit	A numeric specifying the maximum number of iterations that asreml should perform in fitting a model.
IClikelihood	A character that controls both the occurrence and the type of likelihood for information criterion in the test.summary of the new asrtests.object. If none, none are included. Otherwise, if REML, then the AIC and BIC based on the Restricted Maximum Likelihood are included; if full, then the AIC and BIC based on the full likelihood, evaluated using REML estimates, are included. (See also infoCriteria.asreml.)
which.IC	A character specifying the information criterion to be used in selecting the best model. Possible values are AIC and BIC. The value of the criterion for supplied model must exceed that for changed model for the changed model to be returned. (For choosing the rotation angle of the eigenvectors of the penalty matrix, see which.rotacriterion.
...	Further arguments passed to changeModelOnIC.asrtests, newfit.asreml, asreml and tpsmmb.

Details

The model to which the spatial models is to be added is supplied in the asrtests.obj. It should not include terms that will be included in the local spatial model. All spatial model terms are fitted as fixed or random. Consequently, the residual model does not have to be iid. The improvement in the fit resulting from the addition of a spatial model to the supplied model is evaluated. Note that the data must be in the order that corresponds to the residual argument with a variable to the right of another variable changes levels in the data frame faster than those of the other variable e.g. Row:Column implies that all levels for Column in consecutive rows of the data.frame with a single Row level.

For the corr spatial model, the default model is an autocorrelation model of order one (ar1) for each dimension. However, any of the single dimension correlation/variance models from asreml can be specified for each dimension, as can no correlation model for a dimension; the models for the two dimensions can differ. Using a forward selection procedure, a series of models are tried, without removing boundary or singular terms, beginning with the addition of row correlation and followed by the addition of column correlation or, if the row.corrFitfirst is set to FALSE, the reverse order. If the fitting of the first-fitted correlation did not result in a model change because the fitting did not converge or correlations were fixed, but the fit of the second correlation was successful, then adding the first correlation will be retried. If one of the metric correlation functions is specified (e.g. exp), then the row.covar or col.covar will be used in the spatial model. However, because the correlations are fitted separately for the two dimensions, the row.factor and col.factor are needed for all models and is used for a dimension that does not involve a correlation/variance function for the fit being performed. Also, the correlation models are fitted as random terms and so the correlation model will include a variance parameter for the grid even when ar1 is used to specify the correlation model, i.e. the model fitted is a variance model and there is no difference between ar1 and ar1v in fitting the model. The variance parameter for this term represents the spatial variance and the fit necessarily includes a nugget term, this being the residual variance. If any correlation is retained in the model, for a section if sections is not NULL, then the need for a nugget term is assessed by fixing the corresponding residual variance to one, unless there are multiple residual variances and these are not related to the sections. Once the fitting of the correlation model has been completed, the rmboundary function will be executed with the checkboundaryonly value supplied in the addSpatialModel.asrtests call. Finally, checking for bound and singular

random terms associated with the correlation model and residual terms will be carried out when there are correlation terms in the model and checkboundaryonly has been set to FALSE; as many as possible will be removed from the fitted model, in some cases by fixing variance terms to one.

The tensor-product natural-cubic-smoothing-spline (TPNCSS) spatial model is as described by Verbyla et al. (2018), the tensor-product penalized-cubic-spline (TPPSC2) model with second-order differencing of the penalty is similar to that described by Rodriguez-Alvarez et al. (2018), and the tensor-product, first-difference-penalty, linear spline (TPPSL1) model is amongst those described by Piepho, Boer and Williams (2022). The fixed terms for the spline models are row.covar + col.covar + row.covar:col.covar and the random terms are spl(row.covar) + spl(col.covar) + dev(row.covar) + dev(col.covar) + spl(row.covar):col.covar + row.covar:spl(col.covar) + spl(row.covar):spl(col.covar), except that spl(row.covar) + spl(col.covar) is replaced with spl(row.covar):int(col.covar) + int(row.covar):spl(col.covar) in the TPPSC2 model, where int(.) indicates an intercept or constant value specific to its argument. For TPPSL1 models, the terms spl(row.covar):col.covar + row.covar:spl(col.covar) are omitted, The supplied model should not include any of these terms. However, any fixed or random main-effect Row or Column term that has been included as an initial model for comparison with a spatial model can be removed prior to fitting the spatial model using dropFixed or dropRandom. For the P-spline models with second-order differencing, the model matrices used to fit the pairs of random terms (i) spl(row.covar):int(col.covar) and spl(row.covar):col.covar and (ii) int(row.covar):spl(col.covar) and row.covar:spl(col.covar) are transformed using the spectral decomposition of their penalty matrices. An unstructured variance model is tried for each of these pairs. For TPPSC2, it is also possible to optimize the rotation of the null-space eigenvectors of the penalty matrix for each of these random-term pairs (for more information see Piepho, Boer and Williams, 2022). The optimization is achieved either using an optimizer or takes the form of a search over a grid of rotation angles for a reduced model; the fit of the full model with rotation using the optimal rotation angles will be returned.

The TPPCS and TPP1LS models are fitted using functions from the R package TPSbits authored by Sue Welham (2022). There are two methods for supplying the spline basis information produced by tpsmmb to asreml. The grp method adds it to the data.frame supplied in the data argument of the asreml call. The mbf method creates smaller data.frames with the spline basis information in the same environment as the internal function that calls the spline-fitting function. If it is desired to use in a later session, an asreml function, or asrtests function that calls asreml, (e.g. predict.asreml, predictPlus.asreml, or changeTerms.asrtests) on an asreml.object created using mbf terms, then the mbf data.frames will need to be recreated using makeTPPSplineMats.data.frame in the new session, supplying, if there has been rotation of the penalty matrix eigenvectors, the theta values that are returned as the attribute theta.opt of the asreml.obj.

All models utlize the function changeTerms.asrtests to fit the spatial model. Arguments from tpsmmb and changeTerms.asrtests can be supplied in calls to addSpatialModel.asrtests and will be passed on to the relevant function through the ellipses argument (...).

The data for experiment can be divided sections and the same spatial model fitted separately to each. The fit over all of the sections is assessed. For more detail see sections above.

Each combination of a row.coords and a col.coords does not have to specify a single observation; for example, to fit a local spatial model to the main units of a split-unit design, each combination would correspond to a main unit and all subunits of the main unit would have the same combination.

Value

An asrtests.object containing the components (i) asreml.obj, possibly with attribute theta.opt, (ii) wald.tab, and (iii) test.summary for the model that includes the spatial model, unless the spatial model fails to be fitted when allow.unconverged and/or allow.fixedcorrelation is set to FALSE. If the asrtests.object is the result of fitting a TPPCS model with an exploration of the rotation of the eigenvectors of the penalty matrix for the linear components, then the asreml.obj will have an attribute theta.opt that contains the optimal rotation angles of the eigenvectors.

Author(s)

Chris Brien

References

Piepho, H.-P., Boer, M. P., & Williams, E. R. (2022). Two-dimensional P-spline smoothing for spatial analysis of plant breeding trials. *Biometrical Journal*, **64**, 835-857.

Rodriguez-Alvarez, M. X., Boer, M. P., van Eeuwijk, F. A., & Eilers, P. H. C. (2018). Correcting for spatial heterogeneity in plant breeding experiments with P-splines. *Spatial Statistics*, **23**, 52-71.

Verbyla, A. P., De Faveri, J., Wilkie, J. D., & Lewis, T. (2018). Tensor Cubic Smoothing Splines in Designed Experiments Requiring Residual Modelling. *Journal of Agricultural, Biological and Environmental Statistics*, **23(4)**, 478-508.

Welham, S. J. (2022) TPSbits: *Creates Structures to Enable Fitting and Examination of 2D Tensor-Product Splines using ASReml-R.* Version 1.0.0.

See Also

as.asrtests, makeTPPSplineMats.data.frame, addSpatialModelOnIC.asrtests, chooseSpatialModelOnIC.asrtests, changeModelOnIC.asrtests, changeTerms.asrtests, rmboundary.asrtests, testranfix.asrtests, testresidual.asrtests, newfit.asreml, reparamSigDevn.asrtests, changeTerms.asrtests, infoCriteria.asreml

Examples

```
## Not run:

data(Wheat.dat)

#Add row and column covariates
Wheat.dat <- within(Wheat.dat,
                   {
                     cColumn <- dae::as.numfac(Column)
                     cColumn <- cColumn  - mean(unique(cColumn))
                     cRow <- dae::as.numfac(Row)
                     cRow <- cRow - mean(unique(cRow))
                   })

#Fit initial model
current.asr <- asreml(yield ~ Rep + WithinColPairs + Variety,
                      random = ~ Row + Column,
```

```
                      data=Wheat.dat)

#Create an asrtests object, removing boundary terms
current.asrt <- as.asrtests(current.asr, NULL, NULL,
                            label = "Random Row and Column effects")
current.asrt <- rmboundary(current.asrt)

#Create an asrtests object with a P-spline spatial variation model
spatial.asrt <- addSpatialModel(current.asrt, spatial.model = "TPPS",
                                row.covar = "cRow", col.covar = "cColumn",
                                dropRowterm = "Row", dropColterm = "Column",
                                asreml.option = "grp")
infoCriteria(current.asrt$asreml.obj)

#Create an asrtests object with a P-spline spatial variation model
#that includes rotation of the eigenvectors of the penalty matrix
spatial.asrt <- addSpatialModel(current.asrt, spatial.model = "TPPS",
                                row.covar = "cRow", col.covar = "cColumn",
                                dropRowterm = "Row", dropColterm = "Column",
                                rotateX = TRUE,
                                which.rotacriterion = "dev",
                                nrotacores = parallel::detectCores(),
                                asreml.option = "mbf")
infoCriteria(current.asrt$asreml.obj)

## End(Not run)
```

`addSpatialModelOnIC.asrtests` *Uses information criteria to decide whether to add a spatial model to account for local spatial variation.*

Description

Adds either a correlation, two-dimensional tensor-product natural cubic smoothing spline (TP-NCSS), or a two-dimensional tensor-product penalized P-spline model (TPPS) to account for the local spatial variation exhibited by a response variable measured on a potentially irregular grid of rows and columns of the units. The data may be arranged in sections for each of which there is a grid and for which the model is to be fitted separately. Also, the rows and columns of a grid are not necessarily one observational unit wide. The spatial model is only added if the information criterion of the supplied model is decreased with the addition of the local spatial model. For `TPPS` models for which the order of differencing the penalty matrix is two, the improvement in the fit from rotating the eigenvectors of the penalty matrix can be investigated; if there is no improvement, the unrotated fit will be returned.

A row is added for each `section` to the `test.summary` `data.frame` of the `asrtests.object` stating whether or not the new model has been swapped for a model in which the spatial model has been add to the supplied model. Convergence and the occurrence of fixed correlations in fitting the model is checked and a note included in the `action` if there was not. All components of the `asrtests.object` are updated to exhibit the differences between the supplied and the new model, if a spatial model is added.

Usage

```
## S3 method for class 'asrtests'
addSpatialModelOnIC(asrtests.obj, spatial.model = "TPPS",
                    sections = NULL,
                    row.covar = "cRow", col.covar = "cCol",
                    row.factor = "Row", col.factor = "Col",
                    corr.funcs = c("ar1", "ar1"), corr.orders = c(0, 0),
                    row.corrFitfirst = TRUE,
                    allow.corrsJointFit = TRUE, nugget.variance = TRUE,
                    dropFixed = NULL, dropRandom = NULL,
                    nsegs = NULL, nestorder = c(1,1),
                    degree = c(3,3), differorder = c(2,2),
                    usRandLinCoeffs = TRUE,
                    rotateX = FALSE, ngridangles = NULL,
                    which.rotacriterion = "AIC", nrotacores = 1,
                    asreml.option = "grp", tpps4mbf.obj = NULL,
                    allow.unconverged = TRUE, allow.fixedcorrelation = TRUE,
                    checkboundaryonly = FALSE, update = TRUE, trace = FALSE,
                    maxit = 30, IClikelihood = "full", which.IC = "AIC", ...)
```

Arguments

asrtests.obj	An asrtests.object containing the components (i) asreml.obj, (ii) wald.tab, and (iii) test.summary.
spatial.model	A single character string nominating the type of spatial model to fit. Possible values are corr, TPNCSS and TPPS.
sections	A single character string that specifies the name of the column in the data.frame that contains the factor that identifies different sections of the data to which separate spatial models are to be fitted. Note that, for other terms that involve sections in the random formula, there should be separate terms for each level of sections. For example, in a blocked experiment involving multiple sites, there should be the sum of separate terms for the Blocks at each Site i.e. a formula that contains terms like at(Site, i):Block for each site and these are separated by '+'. Otherwise, the combined term (e.g. Site:Block) will impact on the fitting of the local spatial models for the different Sites. Similarly, a separate residual variance for each of the sections should be fitted, unless there is a need to fit a different variance structure to the residual, e.g. heterogeneous residual variances depending on treatments. Separate residual variances for sections can be achieved using the asreml functions dsum or idh. Because, unlike random terms, terms for residual variances are not removed from the model, compound residual terms can be used to include them in the model, e.g. terms with idh or dsum with multiple levels in the list or leaving levels out altogether. In addition to allowing the independent fitting of models to the sections, separate residual variance terms allows a nugget variance to be fitted in a correlation model for each of the sections.
row.covar	A single character string nominating a numeric that contains the values of a centred covariate indexing the rows of a grid. The numeric must be a column in

the data.frame stored in the asreml.obj that is a component of the supplied asrtests.obj.

col.covar A single character string nominating a numeric that contains the values of a centred covariate indexing the columns of a grid. The numeric must be a column in the data.frame stored in the asreml.obj that is a component of the supplied asrtests.obj.

row.factor A single character string nominating a factor that indexes the rows of a grid that are to be one dimension of a spatial correlation model. The factor must a column in the data.frame stored in the asreml.obj that is a component of the supplied asrtests.obj.

col.factor A single character string nominating a factor that indexes the columns of a grid that are to be one dimension of a spatial correlation model. The factor must a column in the data.frame stored in the asreml.obj that is a component of the supplied asrtests.obj.

corr.funcs A single character string of length two that specifies the asreml one-dimensional correlation or variance model function for the row and column dimensions of a two-dimensional separable spatial correlation model to be fitted when spatial.model is corr; the two-dimensional model is fitted as a random term. If a correlation or variance model is not to be investigated for one of the dimensions, specify "" for that dimension. If the correlation model is corb, the values of corr.orders are used for its order argument (b).

corr.orders A numeric of length two that specifies the order argument (b) values for the row and column dimensions of a two-dimensional separable spatial correlation model when spatial.model is corr and the corr.funcs for a dimension is corb, the asreml banded correlation model. If one of the dimensions does not involve an order argument, set the value of corr.orders for that dimension to zero. For a dimension for which the corr.funcs is corb and corr.orders is zero, a model with a single band, the correlation between immediate neighbours, will be fitted and then further bands, up to a maximum of 10 bands, will be added until the addition of an extra band does not reduce the information criterion nominated using which.IC. Note that the two-dimensional spatial model is fitted as a random term.

row.corrFitfirst

A logical. If TRUE then, in fitting the model for spatial.model set to corr, the row correlation or variance function is fitted first, followed by the addition of the column correlation or variance function. If FALSE, the order of fitting is reversed.

allow.corrsJointFit

A logical which, if TRUE, will allow the simultaneous fitting of correlation functions for the two dimensions of the grid when separate fits have failed to fit any correlation functions. This argument is available for when a joint fit hangs the system.

nugget.variance

A logical which, if TRUE, will result in an attempt to fit a nugget or unit-specific variance. Otherwise, a nugget or unit-specific variance will not be fitted.

dropFixed A single character string or a character vector of strings with an element for each level of sections in the same order as the sections levels. Each

string, which if it is not NA and after the addition of ". ~ . -" and conversion to a formula that is then expanded, specifies the sum of a set of terms to be dropped from the fixed formula in fitting splines (TPPS and TPNCSS). The result is that the fitted model supplied in the asrtests.obj, that includes these terms, will be compared with the fitted model that has had them removed and a spatial model added.

An element that is NA indicates that no term pertaining to the corresponding sections level is to be removed. If sections is not NULL and a single character string has been supplied, the terms specified by the string are taken to be terms that are independent of the sections and will be removed when adding the spatial model for the first sections.

The terms must match those in the wald.tab component of the asrtests.obj. The fixed terms will be reordered so that single-variable terms come first, followed by two-variable terms and so on. Note also that multiple terms specified using a single asreml::at function can only be dropped as a whole. If the term was specified using an asreml::at function with a single level, then it can be removed and either the level itself or its numeric position in the levels returned by the levels function can be specified.

dropRandom — A single character string or a character vector of strings with an element for each level of sections in the same order as the sections levels. Each string, which if it is not NA and after the addition of " ~ . -" and conversion to a formula that is then expanded, specifies the sum of a set of terms to be dropped from the random formula in fitting splines (TPPS and TPNCSS). The result is that the fitted model supplied in the asrtests.obj, that includes these terms, will be compared with the fitted model that has had them removed and a spatial model added.

An element that is NA indicates that no term pertaining to the corresponding sections level is to be removed. If sections is not NULL and a single character string has been supplied, the terms specified by the string are taken to be terms that are independent of the sections and will be removed when adding the spatial model for the first sections.

The terms must match those in the vparameters component of the asreml.obj component in the asrtests.obj. Note also that multiple terms specified using a single asreml::at function can only be dropped as a whole. If the term was specified using an asreml::at function with a single level, then it can be removed and either the level itself or its numeric position in the levels returned by the levels function can be specified.

nsegs — A pair of numeric values giving the number of segments into which the column and row ranges are to be split, respectively, for fitting a P-spline model (TPPS) (each value specifies the number of internal knots + 1). If not specified, then (number of unique values - 1) is used in each dimension; for a grid layout with equal spacing, this gives a knot at each data value. If sections is not NULL and the grid differs between the sections, then nsegs will differ between the sections.

nestorder — A numeric of length 2. The order of nesting for column and row dimensions, respectively, in fitting a P-spline model (TPPS). A value of 1 specifies no nesting, a value of 2 generates a spline with half the number of segments in that dimen-

sion, etc. The number of segments in each direction must be a multiple of the order of nesting.

`degree` A `numeric` of length 2. The degree of polynomial spline to be used for column and row dimensions respectively, in fitting a P-spline (`TPPS`).

`differorder` A `numeric` of length 2. The order of differencing for column and row dimensions, respectively, in fitting a P-spline (`TPPS`).

`usRandLinCoeffs`
A `logical` which, if `TRUE` and a P-spline (TPPS) is being fittted, will attempt to fit an unstructured variance model to the constant and linear terms in the interactions for constant and linear terms in one grid dimension interacting with smooth terms in the second grid dimension. The unstructured variance model can only be fitted if both the constant and linear interaction terms have been retained in the fitted model. This argument can be used to omit the attempt to fit an unstructured variance model when the attempt results in a system error.

`rotateX` A `logical` indicating whether to rotate the eigenvectors of the penalty matrix, as described by Piepho, Boer and Williams (2022), when fitting a P-spline (`TPPS`). Setting `rotateX` to `TRUE` results in a search for an optimized rotation under a model that omits the random spline interaction terms. If `ngridangles` is set to `NULL`, the optimal rotation us found using an optimizer (`nloptr::bobyqa`). Otherwise, the optimal rotation is found by exploring the fit over a two-dimensional grid of rotation angle pairs. The optimization seeks to optimize the criterion nominated in `which.rotacriterion`. Rotation of the eigenvectors is only relevant for `differorder` values greater than 1 and has only been implemented for `differorder` equal to 2.

`ngridangles` A `numeric` of length 2. If `NULL` (the default), the optimal pair of angles for rotating the eigenvectors of the penalty matrix of a P-spline (`TPPS`) will be determined using a nonlinear optimizer (`nloptr::bobyqa`). Otherwise, its two values specify the numbers of angles between 0 and 90 degrees for each of the row and column dimensions to be used in determining the optimal pair of angles. Specifying factors of 90 will result in integer-valued angles. The number of grid points, and hence re-analyses will be the product of the values of (`ngridangles` + 1).

`which.rotacriterion`
A single `character` string nominating which of the criteria out of the `deviance`, the `likelihood`, the `AIC` and the `BIC` in determining the optimal rotation of the eigenvectors of the penalty matrix. The `deviance` uses the REML value computed by `asreml`; the other criteria use the full likelihood, evaluated using the REML estimates, that is computed by `infoCriteria.asreml`.

`nrotacores` A `numeric` specifying the number of cores to deploy for running the analyses required to search the two-dimensional grid of rotation angles when `rotateX` is `TRUE`. Parallel processing has been implemented for analyzing, for each column angle, the set of angles to be investigated for the row dimension. The default value of one means that parallel processing will not be used. The value chosen for `nrotacores` needs to balanced against the other processes that are using parallel processing at the same time.

`asreml.option` A single `character` string specifying whether the `grp` or `mbf` methods are to be used to supply externally formed covariate matrices to `asreml` when fit-

ting a P-spline (TPPS). Compared to the mbf method, the grp method is somewhat faster, but creates large asrtests.objects for which the time it takes to save them can exceed any gains in execution speed. The grp method adds columns to the data.frame containing the data. On the other hand, the mbf method adds only the fixed covariates to data and stores the random covariates in the environment of the internal function that calls the spline-fitting function; there are three smaller data.frames for each section that are not stored in the asreml.object resulting from the fitted model.

tpps4mbf.obj
: An object made with makeTPPSplineMats.data.frame that contains the spline basis information for fitting P-splines. The argument tpps4mbf.obj only needs to be set when the mbf option of asreml.option is being used and it is desired to use mbf data.frames that have been created and stored prior to calling addSpatialModelOnIC.asrtests. If tpps4mbf.obj is NULL, makeTPPSplineMats.data.frame will be called internally to produce the required mbf data.frames.

allow.unconverged
: A logical indicating whether to accept a new model even when it does not converge. If FALSE and the fit of the new model does not converge, the supplied asrtests.obj is returned. Also, if FALSE and the fit of the new model has converged, but that of the old model has not, the new model will be accepted.

allow.fixedcorrelation
: A logical indicating whether to accept a new model even when it contains correlations in the model whose values have been designated as fixed, bound or singular. If FALSE and the new model contains correlations whose values have not been able to be estimated, the supplied asrtests.obj is returned. The fit in the asreml.obj component of the supplied asrtests.obj will also be tested and a warning issued if both fixed correlations are found in it and allow.fixedcorrelation is FALSE.

checkboundaryonly
: If TRUE then boundary and singular terms are not removed by rmboundary.asrtests; a warning is issued instead. Note that, for correlation models, the fitting of each dimension and the test for a nugget term are performed with checkboundaryonly set to TRUE and its supplied setting only honoured using a call to rmboundary.asrtests immediately prior to returning the final result of the fitting.

update
: If TRUE, then newfit.asreml is called to fit the model to be tested, using the values of the variance parameters stored in the asreml.object, that is stored in asrtests.obj, as starting values. If FALSE, then newfit.asreml will not use the stored variance parameter values as starting values when fitting the new model, the only modifications being (i) to add the terms for the spatial models and (ii) those specified via ....

trace
: If TRUE then the stages in fitting a correlation model are displayed.

which.IC
: A character specifying the information criterion to be used in selecting the best model. Possible values are AIC and BIC. The value of the criterion for supplied model must exceed that for changed model for the changed model to be returned. (For choosing the rotation angle of the eigenvectors of the penalty matrix, see which.rotacriterion.

maxit
: A numeric specifying the maximum number of iterations that asreml should perform in fitting a model.

ICLikelihood — A character specifying whether Restricted Maximum Likelihood (REML) or the full likelihood, evaluated using REML estimates, (full) are to be used in calculating the information criteria to be included in the test.summary of an asrtests.object or to be used in choosing the best model.

... — Further arguments passed to changeModelOnIC.asrtests, asreml and tpsmmb.

Details

A fitted spatial model is only returned if it improves the fit over and above that of achieved with the model fit supplied in the asrtests.obj. To fit the spatial model without any hypotheses testing or comparison of information criteria use addSpatialModel.asrtests. The model fit supplied in the asrtests.obj should not include terms that will be included in the local spatial model. All spatial model terms are fitted as fixed or random. Consequently, the residual model does not have to be iid. Note that the data must be in the order that corresponds to the residual argument with a variable to the right of another variable changes levels in the data frame faster than those of the other variable e.g. Row:Column implies that all levels for Column in consecutive rows of the data.frame with a single Row level.

For the corr spatial model, the default model is an autocorrelation model of order one (ar1) for each dimension. However, any of the single dimension correlation/variance models from asreml can be specified for each dimension, as can no correlation model for a dimension; the models for the two dimensions can differ. Using a forward selection procedure, a series of models are tried, without removing boundary or singular terms, beginning with the addition of row correlation and followed by the addition of column correlation or, if the row.corrFitfirst is set to FALSE, the reverse order. If the fitting of the first-fitted correlation did not result in a model change because the fitting did not converge or correlations were fixed, but the fit of the second correlation was successful, then adding the first correlation will be retried. If one of the metric correlation functions is specified (e.g. exp), then the row.covar or col.covar will be used in the spatial model. However, because the correlations are fitted separately for the two dimensions, the row.factor and col.factor are needed for all models and is used for a dimension that does not involve a correlation/variance function for the fit being performed. Also, the correlation models are fitted as random terms and so the correlation model will include a variance parameter for the grid even when ar1 is used to specify the correlation model, i.e. the model fitted is a variance model and there is no difference between ar1 and ar1v in fitting the model. The variance parameter for this term represents the spatial variance and the fit necessarily includes a nugget term, this being the residual variance. If any correlation is retained in the model, for a section if sections is not NULL, then the need for a nuggest term is assessed by fixing the corresponding residual variance to one, unless there are multiple residual variances and these are not related to the sections. Once the fitting of the correlation model has been completed, the rmboundary function will be executed with the checkboundaryonly value supplied in the addSpatialModelOnIC.asrtests call. Finally, checking for bound and singular random terms associated with the correlation model and residual terms will be carried out when there are correlation terms in the model and checkboundaryonly has been set to FALSE; as many as possible will be removed from the fitted model, in some cases by fixing variance terms to one.

The tensor-product natural-cubic-smoothing-spline (TPNCSS) spatial model is as described by Verbyla et al. (2018), the tensor-product penalized-cubic-spline (TPPSC2) model with second-order differencing of the penalty is similar to that described by Rodriguez-Alvarez et al. (2018), and the tensor-product, first-difference-penalty, linear spline (TPPSL1) model is amongst those described by Piepho, Boer and Williams (2022). The fixed terms for the spline models are row.covar + col.covar + row.covar:col.covar and the random terms are spl(row.covar) + spl(col.covar)

+ dev(row.covar) + dev(col.covar) + spl(row.covar):col.covar + row.covar:spl(col.covar) + spl(row.covar):spl(col.covar), except that spl(row.covar) + spl(col.covar) is replaced with spl(row.covar):int(col.covar) + int(row.covar):spl(col.covar) in the TPPSC2 model, where int(.) indicates an intercept or constant value specific to its argument. For TPPSL1 models, the terms spl(row.covar):col.covar + row.covar:spl(col.covar) are omitted, The supplied model should not include any of these terms. However, any fixed or random main-effect Row or Column term that has been included as an initial model for comparison with a spatial model can be removed prior to fitting the spatial model using dropFixed or dropRandom. For the P-spline models with second-order differencing, the model matrices used to fit the pairs of random terms (i) spl(row.covar):int(col.covar) and spl(row.covar):col.covar and (ii) int(row.covar):spl(col.covar) and row.covar:spl(col.covar) are transformed using the spectral decomposition of their penalty matrices. An unstructured variance model is tried for each of these pairs and retained if it improves the fit. For TPPSC2, it is also possible to optimize the rotation of the null-space eigenvectors of the penalty matrix for each of these random-term pairs (for more information see Piepho, Boer and Williams, 2022). The optimization is achieved either using an optimizer or takes the form of a search over a grid of rotation angles for a reduced model; the fit of the full model with rotation using the optimal rotation angles will only be returned if it improves on the fit of the full, unrotated model.

The TPPCS and TPP1LS models are fitted using functions from the R package TPSbits authored by Sue Welham (2022). There are two methods for supplying the spline basis information produced by tpsmmb to asreml. The grp method adds it to the data.frame supplied in the data argument of the asreml call. The mbf method creates smaller data.frames with the spline basis information in the same environment as the internal function that calls the spline-fitting function. If it is desired to use in a later session, an asreml function, or asrtests function that calls asreml, (e.g. predict.asreml, predictPlus.asreml, or changeTerms.asrtests) on an asreml.object created using mbf terms, then the mbf data.frames will need to be recreated using makeTPPSplineMats.data.frame in the new session, supplying, if there has been rotation of the penalty matrix eigenvectors, the theta values that are returned as the attribute theta.opt of the asreml.obj.

All models utlize the function changeModelOnIC.asrtests to assess the model fit, the information criteria used in assessing the fit being calculated using infoCriteria. Any bound terms are removed from the model. Arguments from tpsmmb and changeModelOnIC.asrtests can be supplied in calls to addSpatialModelOnIC.asrtests and will be passed on to the relevant function through the ellipses argument (...).

The data for experiment can be divided sections and the same spatial model fitted separately to each. The fit over all of the sections is assessed. For more detail see sections above.

Each combination of a row.coords and a col.coords does not have to specify a single observation; for example, to fit a local spatial model to the main units of a split-unit design, each combination would correspond to a main unit and all subunits of the main unit would have the same combination.

Value

An asrtests.object containing the components (i) asreml.obj, possibly with attribute theta.opt, (ii) wald.tab, and (iii) test.summary for the model whose fit has the smallest information criterion between the supplied and spatial model. The values of the degrees of freedom and the information criteria in the test.summary are differences between those of the changed model and those of the model supplied to addSpatialModelOnIC. If the asrtests.object is the result of fitting a TPPCS

model with an exploration of the rotation of the eigenvectors of the penalty matrix for the linear components, then the `asreml.obj` will have an attribute `theta.opt` that contains the optimal rotation angles of the eigenvectors.

Author(s)

Chris Brien

References

Piepho, H.-P., Boer, M. P., & Williams, E. R. (2022). Two-dimensional P-spline smoothing for spatial analysis of plant breeding trials. *Biometrical Journal*, **64**, 835-857.

Rodriguez-Alvarez, M. X., Boer, M. P., van Eeuwijk, F. A., & Eilers, P. H. C. (2018). Correcting for spatial heterogeneity in plant breeding experiments with P-splines. *Spatial Statistics*, **23**, 52-71.

Verbyla, A. P., De Faveri, J., Wilkie, J. D., & Lewis, T. (2018). Tensor Cubic Smoothing Splines in Designed Experiments Requiring Residual Modelling. *Journal of Agricultural, Biological and Environmental Statistics*, **23(4)**, 478-508.

Welham, S. J. (2022) `TPSbits`: *Creates Structures to Enable Fitting and Examination of 2D Tensor-Product Splines using ASReml-R*. Version 1.0.0.

See Also

`as.asrtests`, `makeTPPSplineMats.data.frame`, `addSpatialModel.asrtests`, `chooseSpatialModelOnIC.asrtests`, `changeModelOnIC.asrtests`, `changeTerms.asrtests`, `rmboundary.asrtests`, `testranfix.asrtests`, `testresidual.asrtests`, `newfit.asreml`, `reparamSigDevn.asrtests`, `changeTerms.asrtests`, `infoCriteria.asreml`

Examples

```
## Not run:

data(Wheat.dat)

#Add row and column covariates
Wheat.dat <- within(Wheat.dat,
                    {
                      cColumn <- dae::as.numfac(Column)
                      cColumn <- cColumn  - mean(unique(cColumn))
                      cRow <- dae::as.numfac(Row)
                      cRow <- cRow - mean(unique(cRow))
                    })

#Fit initial model
current.asr <- asreml(yield ~ Rep + WithinColPairs + Variety,
                      random = ~ Row + Column,
                      data=Wheat.dat)

#Create an asrtests object, removing boundary terms
current.asrt <- as.asrtests(current.asr, NULL, NULL,
                            label = "Random Row and Column effects")
```

```
current.asrt <- rmboundary(current.asrt)

current.asrt <- addSpatialModelOnIC(current.asrt, spatial.model = "TPPS",
                                    row.covar = "cRow", col.covar = "cColumn",
                                    dropRowterm = "Row", dropColterm = "Column",
                                    asreml.option = "grp")
infoCriteria(current.asrt$asreml.obj)

## End(Not run)
```

`addto.test.summary` *Adds a row to a* `test.summary` `data.frame`.

Description

A row that summarizes the result of a proposed change to a model is added to a `test.summary` `data.frame`. Only the values of those arguments for which there are columns in `test.summary` will be included in the row.

Usage

```
addto.test.summary(test.summary, terms, DF = 1, denDF = NA,
                           p = NA, AIC = NA, BIC = NA,
                           action = "Boundary")
```

Arguments

`test.summary`	A `data.frame` whose columns are a subset of `terms`, `DF`, `denDF`, `p`, `AIC`, `BIC` and `action`. Each row summarizes the results of proposed changes to the fitted model. See `asrtests.object` for more information.
`terms`	A `character` giving the name of a term that might be added to or removed from the model or a label indicating a change that might be made to the model.
`DF`	A `numeric` giving the numerator degrees of freedom for a Wald F-statistic or the number of variance parameters in the current model minus the number in the proposed model.
`denDF`	A `numeric` giving the denominator degrees of freedom for a Wald F-statistic.
`p`	A `numeric` giving the p-value for a Wald F-statistic or REML ratio test.
`AIC`	A `numeric` giving Akiake Information Criterion (AIC) for a model or the difference between the AIC values for the current and proposed models.
`BIC`	A `numeric` giving Bayesian (Schwarz) Information Criterion for a model or the difference between the AIC values for the current and proposed models.
`action`	A `character` giving what action was taken with respect to the proposed change. See `asrtests.object` for more information.

Value

A `data.frame`.

Author(s)

Chris Brien

See Also

asremlPlus-package, asrtests.object, print.test.summary

Examples

```
  ## Not run:
data(Wheat.dat)

## Fit an autocorrelation model
ar1.asr <- asreml(yield ~ Rep + WithinColPairs + Variety,
                  random = ~ Row + Column + units,
                  residual = ~ ar1(Row):ar1(Column),
                  data=Wheat.dat)
ar1.asrt <- as.asrtests(ar1.asr, NULL, NULL,
                        label = "Autocorrelation model")
ar1.asrt <- rmboundary.asrtests(ar1.asrt)

## Fit a tensor spline
Wheat.dat <- within(Wheat.dat,
                    {
                      cRow <- dae::as.numfac(Row)
                      cRow <- cRow - mean(unique(cRow))
                      cColumn <- dae::as.numfac(Column)
                      cColumn <- cColumn - mean(unique(cColumn))
                    })
ts.asr <- asreml(yield ~ Rep + cRow + cColumn + WithinColPairs +
                         Variety,
                  random = ~ spl(cRow) + spl(cColumn) +
                             dev(cRow) + dev(cColumn) +
                             spl(cRow):cColumn + cRow:spl(cColumn) +
                             spl(cRow):spl(cColumn),
                  residual = ~ Row:Column,
                  data=Wheat.dat)
ts.asrt <- as.asrtests(ts.asr, NULL, NULL,
                       label = "Tensor spline model")
ts.asrt <- rmboundary.asrtests(ts.asrt)
ar1.ic <- infoCriteria(ar1.asrt$asreml.obj)
ts.ic <- infoCriteria(ts.asrt$asreml.obj)
if (ar1.ic$AIC < ts.ic$AIC)
{
  ic.diff <- ar1.ic - ts.ic
  new.asrt <- ar1.asrt
  new.asrt$test.summary <- addto.test.summary(ar1.asrt$test.summary,
                                              terms = "Compare ar1 to ts",
                                              DF = ic.diff$varDF,
                                              AIC = ic.diff$AIC, BIC = ic.diff$BIC,
                                              action = "Chose ar1")
} else
```

```
{
  ic.diff <- ts.ic - ar1.ic
  new.asrt <- ts.asrt
  new.asrt$test.summary <- addto.test.summary(ts.asrt$test.summary,
                                              terms = "Compare ar1 to ts",
                                              DF = ic.diff$varDF,
                                              AIC = ic.diff$AIC, BIC = ic.diff$BIC,
                                              action = "Chose ts")
}

## End(Not run)
```

`allDifferences.data.frame` *Using supplied predictions and standard errors of pairwise differences or the variance matrix of predictions, forms all pairwise differences between the set of predictions, and p-values for the differences.*

Description

Uses supplied predictions and standard errors of pairwise differences, or the variance matrix of predictions to form, in an `alldiffs.object`, for those components not already present, (i) a table of all pairwise differences of the predictions, (ii) the p-value of each pairwise difference, and (iii) the minimum, mean, maximum and accuracy of LSD values. Predictions that are aliased (or inestimable) are removed from the `predictions` component of the `alldiffs.object` and standard errors of differences involving them are removed from the `sed` component.

If necessary, the order of the columns of the variables in the `predictions` component are changed to be the initial columns of the `predictions.frame` and to match their order in the `classify`. Also, the rows of `predictions` component are ordered so that they are in standard order for the variables in the `classify`. That is, the values of the last variable change with every row, those of the second-last variable only change after all the values of the last variable have been traversed; in general, the values of a variable are the same for all the combinations of the values to the variables to its right in the `classify`. The `sortFactor` or `sortOrder` arguments can be used to order of the values for the `classify` variables, which is achieved using `sort.alldiffs`.

Each p-value is computed as the probability of a t-statistic as large as or larger than the absolute value of the observed difference divided by its standard error. The p-values are stored in the `p.differences` component. The degrees of freedom of the t-distribution is the degrees of freedom stored in the `tdf` attribute of the `alldiffs.object`. This t-distribution is also used in calculating the LSD statistics stored in the `LSD` component of the `alldiffs.object`.

Usage

```
## S3 method for class 'data.frame'
allDifferences(predictions, classify, vcov = NULL,
               differences = NULL, p.differences = NULL, sed = NULL,
               LSD = NULL, LSDtype = "overall", LSDsupplied = NULL,
               LSDby = NULL, LSDstatistic = "mean",
```

```
LSDaccuracy = "maxAbsDeviation",
retain.zeroLSDs = FALSE,
zero.tolerance = .Machine$double.eps ^ 0.5,
backtransforms = NULL,
response = NULL, response.title = NULL,
term = NULL, tdf = NULL,
x.num = NULL, x.fac = NULL,
level.length = NA,
pairwise = TRUE, alpha = 0.05,
transform.power = 1, offset = 0, scale = 1,
transform.function = "identity",
inestimable.rm = TRUE,
sortFactor = NULL, sortParallelToCombo = NULL,
sortNestingFactor = NULL, sortOrder = NULL,
decreasing = FALSE, ...)
```

Arguments

predictions	A predictions.frame, or a data.frame, beginning with the variables classifying the predictions and also containing columns named predicted.value, standard.error and est.status; each row contains a single predicted value. It may also contain columns for the lower and upper limits of error intervals for the predictions. Note that the names standard.error and est.status have been changed to std.error and status in the pvals component produced by asreml-R4; if the new names are in the data.frame supplied to predictions, they will be returned to the previous names.
classify	A character string giving the variables that define the margins of the multiway table that has been predicted. Multiway tables are specified by forming an interaction type term from the classifying variables, that is, separating the variable names with the : operator.
vcov	A matrix containing the variance matrix of the predictions; it is used in computing the variance of linear transformations of the predictions.
differences	A matrix containing all pairwise differences between the predictions; it should have the same number of rows and columns as there are rows in predictions.
p.differences	A matrix containing p-values for all pairwise differences between the predictions; each p-value is computed as the probability of a t-statistic as large as or larger than the observed difference divided by its standard error. The degrees of freedom of the t distribution for computing it are computed as the denominator degrees of freedom of the F value for the fixed term, if available; otherwise, the degrees of freedom stored in the attribute tdf are used; the matrix should be of the same size as that for differences.
sed	A matrix containing the standard errors of all pairwise differences between the predictions; they are used in computing the p-values.
LSD	An LSD.frame containing the mean, minimum and maximum LSD for determining the significance of pairwise differences, as well as an assigned LSD and a measure of the accuracy of the LSD. If LSD is NULL then the LSD.frame stored in the LSD component will be calculated and the values of LSDtype, LSDby and

LSDstatistic added as attributes of the alldiffs.object. The LSD for a single prediction assumes that any predictions to be compared are independent; this is not the case if residual errors are correlated.

LSDtype
: A character string that can be overall, factor.combinations, per.prediction or supplied. It determines whether the values stored in a row of a LSD.frame are the values calculated (i) overall from the LSD values for all pairwise comparisons, unless there is only one, possibly repeated, prediction, when a notional LSD is calculated, (ii) the values calculated from the pairwise LSDs for the levels of each factor.combination, unless there is only one prediction for a level of the factor.combination, when a notional LSD is calculated, (iii) per.prediction, being based, for each prediction, on all pairwise differences involving that prediction, or (iv) as supplied values of the LSD, specified with the LSDsupplied argument; these supplied values are to be placed in the assignedLSD column of the LSD.frame stored in an alldiffs.object so that they can be used in LSD calculations.

  See LSD.frame for further information on the values in a row of this data.frame and how they are calculated.

LSDsupplied
: A data.frame or a named numeric containing a set of LSD values that correspond to the observed combinations of the values of the LSDby variables in the predictions.frame or a single LSD value that is an overall LSD. If a data.frame, it may have (i) a column for the LSDby variable and a column of LSD values or (ii) a single column of LSD values with rownames being the combinations of the observed values of the LSDby variables. Any name can be used for the column of LSD values; assignedLSD is sensible, but not obligatory. Otherwise, a numeric containing the LSD values, each of which is named for the observed combination of the values of the LSDby variables to which it corresponds. (Applying the function dae::fac.combine to the predictions component is one way of forming the required combinations for the (row) names.) The values supplied will be incorporated into assignedLSD column of the LSD.frame stored as the LSD component of the alldiffs.object.

LSDby
: A character (vector) of variables names, being the names of the factors or numerics in the classify; for each combination of their levels and values, there will be or is a row in the LSD.frame stored in the LSD component of the alldiffs.object when LSDtype is factor.combinatons.

LSDstatistic
: A character nominating one or more of minimum, q10, q25, mean, median, q75, q90 or maximum as the value(s) to be stored in the assignedLSD column in an LSD.frame; the values in the assignedLSD column are used in computing halfLeastSignificant error.intervals. Here q10, q25, q75 and q90 indicate the sample quantiles corresponding to probabilities of 0.1, 0.25, 0.75 and 0.9 for the group of LSDs from which a single LSD value is calculated. The function quantile is used to obtain them. The mean LSD is calculated as the square root of the mean of the squares of the LSDs for the group. The median is calculated using the median function. Multiple values are only produced for LSDtype set to factor.combination, in which case LSDby must not be NULL and the number of values must equal the number of observed combinations of the values of the variables specified by LSDby. If LSDstatistic is NULL, it is reset to mean.

LSDaccuracy A character nominating one of maxAbsDeviation, maxDeviation, q90Deviation or RootMeanSqDeviation as the statistic to be calculated as a measure of the accuracy of assignedLSD. The option q90Deviation produces the sample quantile corresponding to a probability of 0.90. The deviations are the differences between the LSDs used in calculating the LSD statistics and each assigned LSD and the accuracy is expressed as a proportion of the assigned LSD value. The calculated values are stored in the column named accuracyLSD in an LSD.frame.

retain.zeroLSDs A logical indicating whether to retain or omit LSDs that are zero when calculating the summaries of LSDs.

zero.tolerance A numeric specifying the value such that if an LSD is less than it, the LSD will be considered to be zero.

backtransforms A data.frame containing the backtransformed values of the predicted values that is consistent with the predictions component, except that the column named predicted.value is replaced by one called backtransformed.predictions. Any error.interval values will also be the backtransformed values. Each row contains a single predicted value.

response A character specifying the response variable for the predictions. It is stored as an attribute to the alldiffs.object.

response.title A character specifying the title for the response variable for the predictions. It is stored as an attribute to the alldiffs.object.

term A character string giving the variables that define the term that was fitted using asreml and that corresponds to classify. It only needs to be specified when it is different to classify; it is stored as an attribute of the alldiffs.object. It is likely to be needed when the fitted model includes terms that involve both a numeric covariate and a factor that parallel each other; the classify would include the covariate and the term would include the factor.

tdf an integer specifying the degrees of freedom of the standard error. It is used as the degrees of freedom for the t-distribution on which p-values and confidence intervals are based. It is stored as an attribute to the alldiffs.object.

x.num A character string giving the name of the numeric covariate that (i) is potentially included in terms in the fitted model and (ii) is the x-axis variable for plots. Its values will not be converted to a factor.

x.fac A character string giving the name of the factor that (i) corresponds to x.num and (ii) is potentially included in terms in the fitted model. It should have the same number of levels as the number of unique values in x.num. The levels of x.fac must be in the order in which they are to be plotted - if they are dates, then they should be in the form yyyymmdd, which can be achieved using as.Date. However, the levels can be non-numeric in nature, provided that x.num is also set.

level.length The maximum number of characters from the levels of factors to use in the row and column labels of the tables of pairwise differences and their p-values and standard errors.

pairwise A logical indicating whether all pairwise differences of the predictions and their standard errors and p-values are to be computed and stored. If FALSE,

the components differences and p.differences will be NULL in the returned alldiffs.object.

alpha A numeric giving the significance level for LSDs or one minus the confidence level for confidence intervals. It is stored as an attribute to the alldiffs.object.

transform.power
A numeric specifying the power of a transformation, if one has been applied to the response variable. Unless it is equal to 1, the default, back-transforms of the predictions will be obtained and presented in tables or graphs as appropriate. The back-transformation raises the predictions to the power equal to the reciprocal of transform.power, unless it equals 0 in which case the exponential of the predictions is taken.

offset A numeric that has been added to each value of the response after any scaling and before applying any power transformation.

scale A numeric by which each value of the response has been multiplied before adding any offset and applying any power transformation.

transform.function
A character giving the name of a function that specifies the scale on which the predicted values are defined. This may be the result of a transformation of the data using the function or the use of the function as a link function in the fitting of a generalized linear (mixed) model (GL(M)M). The possible transform.functions are identity, log, inverse, sqrt, logit, probit, and cloglog. The predicted.values and error.intervals, if not StandardError intervals, will be back-transformed using the inverse function of the transform.function. The standard.error column will be set to NA, unless (i) asreml returns columns named transformed.value and approx.se, as well as those called predicted.values and standard.error (such as when a GLM is fitted) and (ii) the values in transformed.value are equal to those obtained by backtransforming the predicted.values using the inverse function of the transform.function. Then, the approx.se values will be saved in the standard.error column of the backtransforms component of the returned alldiffs.obj. Also, the transformed.value and approx.se columns are removed from both the predictions and backtransforms components of the alldiffs.obj. Note that the values that end up in the standard errors column are approximate for the backtransformed values and are not used in calculating error.intervals.

inestimable.rm A logical indicating whether rows for predictions that are not estimable are to be removed from the components of the alldiffs.object.

sortFactor A character containing the name of the factor that indexes the set of predicted values that determines the sorting of the components. If there is only one variable in the classify term then sortFactor can be NULL and the order is defined by the complete set of predicted values. If there is more than one variable in the classify term then sortFactor must be set. In this case the sortFactor is sorted in the same order within each combination of the values of the sortParallelToCombo variables: the classify variables, excluding the sortFactor. There should be only one predicted value for each unique value of sortFactor within each set defined by a combination of the values of the classify variables, excluding the sortFactor factor. The order to use is determined by either sortParallelToCombo or sortOrder.

sortParallelToCombo
: A list that specifies a combination of the values of the factors and numerics, excluding sortFactor, that are in classify. Each of the components of the supplied list is named for a classify variable and specifies a single value for it. The combination of this set of values will be used to define a subset of the predicted values whose order will define the order of sortFactor. Each of the other combinations of the values of the factors and numerics will be sorted in parallel. If sortParallelToCombo is NULL then the first value of each classify variable, except for the sortFactor factor, in the predictions component is used to define sortParallelToCombo. If there is only one variable in the classify then sortParallelToCombo is ignored.

sortNestingFactor
: A character containing the name of the factor that defines groups of the sortFactor within which the predicted values are to be ordered. If there is only one variable in the classify then sortNestingFactor is ignored.

sortOrder
: A character vector whose length is the same as the number of levels for sortFactor in the predictions component of the alldiffs.object. It specifies the desired order of the levels in the reordered components of the alldiffs.object. The argument sortParallelToCombo is ignored.

  The following creates a sortOrder vector levs for factor f based on the values in x:
  levs <- levels(f)[order(x)].

decreasing
: A logical passed to order that detemines whether the order for sorting the components of the alldiffs.object is for increasing or decreasing magnitude of the predicted values.

...
: provision for passsing arguments to functions called internally - not used at present.

Value

An alldiffs.object with components predictions, vcov, differences, p.differences sed, and LSD.

The name of the response, the response.title, the term, the classify, tdf, alpha, sortFactor and the sortOrder will be set as attributes to the object. Note that the classify in an alldiffs.object is based on the variables indexing the predictions, which may differ from the classify used to obtain the original predictions (for example, when the alldiffs.objects stores a linear transformation of predictions.

Also, see predictPlus.asreml for more information.

Author(s)

Chris Brien

See Also

asremlPlus-package, as.alldiffs, as.predictions.frame, sort.alldiffs, subset.alldiffs, print.alldiffs, renewClassify.alldiffs, redoErrorIntervals.alldiffs,

recalcLSD.alldiffs, pickLSDstatistics.alldiffs, plotPredictions.data.frame, predictPlus.asreml, predictPresent.asreml

Examples

```
  data(Oats.dat)

  ## Use asreml to get predictions and associated statistics

  ## Not run:
  m1.asr <- asreml(Yield ~ Nitrogen*Variety,
                   random=~Blocks/Wplots,
                   data=Oats.dat)
  current.asrt <- as.asrtests(m1.asr)
  Var.pred <- asreml::predict.asreml(m1.asr, classify="Nitrogen:Variety",
                                     sed=TRUE)
  if (getASRemlVersionLoaded(nchar = 1) == "3")
    Var.pred <- Var.pred$predictions
  Var.preds <- Var.pred$pvals
  Var.sed <- Var.pred$sed
  Var.vcov <- NULL
  wald.tab <-  current.asrt$wald.tab
  den.df <- wald.tab[match("Variety", rownames(wald.tab)), "denDF"]

## End(Not run)

  ## Use lmerTest and emmmeans to get predictions and associated statistics
  if (requireNamespace("lmerTest", quietly = TRUE) &
      requireNamespace("emmeans", quietly = TRUE))
  {
    m1.lmer <- lmerTest::lmer(Yield ~ Nitrogen*Variety + (1|Blocks/Wplots),
                              data=Oats.dat)
    Var.emm <- emmeans::emmeans(m1.lmer, specs = ~ Nitrogen:Variety)
    Var.preds <- summary(Var.emm)
    den.df <- min(Var.preds$df)
    ## Modify Var.preds to be compatible with a predictions.frame
    Var.preds <- as.predictions.frame(Var.preds, predictions = "emmean",
                                      se = "SE", interval.type = "CI",
                                      interval.names = c("lower.CL", "upper.CL"))
    Var.vcov <- vcov(Var.emm)
    Var.sed <- NULL
  }

  ## Use the predictions obtained with either asreml or lmerTest
  if (exists("Var.preds"))
  {
    ## Order the Varieties in decreasing order for the predictions values in the
    ## first N level
    Var.diffs <- allDifferences(predictions = Var.preds,
                                classify = "Nitrogen:Variety",
                                sed = Var.sed, vcov = Var.vcov, tdf = den.df,
                                sortFactor = "Variety", decreasing = TRUE)
    print.alldiffs(Var.diffs, which="differences")
```

```
  ## Change the order of the factors in the alldiffs object and reorder components
  Var.reord.diffs <- allDifferences(predictions = Var.preds,
                                    classify = "Variety:Nitrogen",
                                    sed = Var.sed, vcov = Var.vcov, tdf = den.df)
  print.alldiffs(Var.reord.diffs, which="predictions")
}
```

---

alldiffs.object *Description of an alldiffs object*

Description

An object of S3-class alldiffs that stores the predictions for a model, along with supplied statistics for all pairwise differences. While alldiffs.object can be constructed by defining a list with the appropriate components, it can be formed by passing the components to as.alldiffs, or from a predictions data.frame using allDifferences.data.frame.

as.alldiffs is function that assembles an object of this class from supplied components.

is.alldiffs is the membership function for this class; it tests that an object is of class alldiffs.

validAlldiffs(object) can be used to test the validity of an object with this class.

allDifferences.data.frame is the function that constructs an object of this class by calculating components from statistics supplied via its arguments and then using as.alldiffs to make the object.

Value

A list with classes alldiffs and list containing the following components: predictions, vcov, differences, p.differences, sed, LSD and backtransforms. Except for predictions, the components are optional and can be set to NULL.

An alldiffs.object also has attributes response, response.title, term, classify, tdf, alpha, sortFactor and sortOrder, which may be set to NULL.

The details of the components are as follows:

1. predictions: A predictions.frame, being a data.frame beginning with the variables classifying the predictions, in the same order as in the classify, and also containing columns named predicted.value, standard.error and est.status; each row contains a single predicted value. The number of rows should equal the number of unique combinations of the classify variables and will be in standard order for the classify variables. That is, the values of the last variable change with every row, those of the second-last variable only change after all the values of the last variable have been traversed; in general, the values of a variable are the same for all the combinations of the values to the variables to its right in the classify.

   The data.frame may also include columns for the lower and upper values of error intervals, either standard error, confidence or half-LSD intervals. The names of these columns will consist of three parts separated by full stops: 1) the first part will be lower or upper; 2) the second part will be one of Confidence, StandardError or halfLeastSignificant; 3) the third component will be limits.

Note that the names standard.error and est.status have been changed to std.error and status in the pvals component produced by asreml-R4; if the new names are in the data.frame supplied to predictions, they will be returned to the previous names.

2. differences: A matrix containing all pairwise differences between the predictions; it should have the same number of rows and columns as there are rows in predictions.
3. p.differences: A matrix containing p-values for all pairwise differences between the predictions; each p-value is computed as the probability of a t-statistic as large as or larger than the observed difference divided by its standard error. The degrees of freedom of the t distribution for computing it are computed as the denominator degrees of freedom of the F value for the fixed term, if available; otherwise, the degrees of freedom stored in the attribute tdf are used; the matrix should be of the same size as that for differences.
4. sed: A matrix containing the standard errors of all pairwise differences between the predictions; they are used in computing the p-values in p.differences.
5. vcov: A matrix containing the variance matrix of the predictions; it is used in computing the variance of linear transformations of the predictions.
6. LSD: An LSD.frame containing (i) c, the number of pairwise predictions comparisons for each LSD value and the mean, minimum, maximum and assigned LSD, (ii) the column accuracyLSD that gives a measure of the accuracy of the assigned LSD. given the variation in LSD values, and (iii) the columns false.pos and false.neg that contain the number of false positives and negatives if the assignedLSD value(s) is(are) used to determine the significance of the pairwise predictions differences. The LSD values in the assignedLSD column is used to determine the significance of pairwise differences that involve predictions for the combination of levels given by a row name. The value in the assignedLSD column is specified using the LSDstatistic argument.
7. backtransforms: When the response values have been transformed for analysis, a data.frame containing the backtransformed values of the predicted values is added to the alldiffs.object. This data.frame is consistent with the predictions component, except that the column named predicted.value is replaced by one called backtransformed.predictions. Any error.interval values will also be the backtransformed values. Each row contains a single predicted value.

The details of the attributes of an alldiffs.object are:

1. response: A character specifying the response variable for the predictions.
2. response.title: A character specifying the title for the response variable for the predictions.
3. term: A character giving the variables that define the term that was fitted using asreml and that corresponds to classify. It is often the same as classify.
4. classify: A character giving the variables that define the margins of the multiway table used in the prediction. Multiway tables are specified by forming an interaction type term from the classifying variables, that is, separating the variable names with the : operator.
5. tdf: An integer specifying the degrees of freedom of the standard error. It is used as the degrees of freedom for the t-distribution on which p-values and confidence intervals are based.
6. alpha: An integer specifying the significance level. It is used as the significance level calculating LSDs.

7. LSDtype: If the LSD component is not NULL then LSDtype is added as an attribute. A character nominating the type of grouping of seds to be used in combining LSDs.
8. LSDby: If the LSD component is not NULL then LSDby is added as an attribute. A character vector containing the names of the factors and numerics within whose combinations the LSDs are to be summarized.
9. LSDstatistic: If the LSD component is not NULL then LSDstatistic is added as an attribute. A character nominating what statistic to use in summarizing a set of LSDs.
10. LSDaccuracy: If the LSD component is not NULL then LSDaccuracy is added as an attribute. A character nominating the method of calculating a measure of the accuracy of the LSDs stored in the assignedLSD column of the LSD.frame.
11. sortFactor: factor that indexes the set of predicted values that determined the sorting of the components.
12. sortOrder: A character vector that is the same length as the number of levels for sortFactor in the predictions component of the alldiffs.object. It specifies the order of the levels in the reordered components of the alldiffs.object.

The following creates a sortOrder vector levs for factor f based on the values in x: levs <- levels(f)[order(x)].

See predictPlus.asreml for more information.

Author(s)

Chris Brien

See Also

is.alldiffs, as.alldiffs, validAlldiffs, allDifferences.data.frame

Examples

```
data(Oats.dat)

## Use asreml to get predictions and associated statistics

## Not run:
m1.asr <- asreml(Yield ~ Nitrogen*Variety,
                 random=~Blocks/Wplots,
                 data=Oats.dat)
current.asrt <- as.asrtests(m1.asr)
Var.pred <- asreml::predict.asreml(m1.asr, classify="Nitrogen:Variety",
                                   sed=TRUE)
if (getASRemlVersionLoaded(nchar = 1) == "3")
  Var.pred <- Var.pred$predictions
Var.preds <- Var.pred$pvals
Var.sed <- Var.pred$sed
Var.vcov <- NULL

## End(Not run)
```

```
## Use lmerTest and emmmeans to get predictions and associated statistics
if (requireNamespace("lmerTest", quietly = TRUE) &
    requireNamespace("emmeans", quietly = TRUE))
{
  m1.lmer <- lmerTest::lmer(Yield ~ Nitrogen*Variety + (1|Blocks/Wplots),
                            data=Oats.dat)
  Var.emm <- emmeans::emmeans(m1.lmer, specs = ~ Nitrogen:Variety)
  Var.preds <- summary(Var.emm)
  den.df <- min(Var.preds$df)
  ## Modify Var.preds to be compatible with a predictions.frame
  Var.preds <- as.predictions.frame(Var.preds, predictions = "emmean",
                                    se = "SE", interval.type = "CI",
                                    interval.names = c("lower.CL", "upper.CL"))
  Var.vcov <- vcov(Var.emm)
  Var.sed <- NULL
}

## Use the predictions obtained with either asreml or lmerTest
if (exists("Var.preds"))
{
  ## Form an all.diffs object
   Var.diffs <- as.alldiffs(predictions = Var.preds, classify = "Nitrogen:Variety",
                            sed = Var.sed, vcov = Var.vcov, tdf = den.df)

  ## Check the class and validity of the alldiffs object
  is.alldiffs(Var.diffs)
  validAlldiffs(Var.diffs)
}
```

`angular` *Applies the angular transformation to proportions.*

Description

Applies the angular transformation to numeric values. It is given by $\sin^{-1}(\sqrt{proportions})$

Usage

```
angular(proportions, n)
```

Arguments

`proportions`	The proportions.
`n`	The divisor(s) for each proportion

Value

A numeric.

Author(s)

Chris Brien

See Also

angular.mod, powerTransform.

Examples

```
n <-25
y <- rbinom(10, n, 0.5)
y <- c(y,0,n)
p <- y/n
p.ang <- angular(p, n)
```

---

angular.mod *Applies the modified angular transformation to a vector of counts.*

Description

Applies the angular transformation to a vector of counts. A modified transformation is used that is appropriate when N < 50 and the proportion is not between 0.3 and 0.7. The transformation is given by $\sin^{-1} \frac{count+0.375}{n+0.75}$.

Usage

```
angular.mod(count, n)
```

Arguments

count	The numeric vector of counts.
n	The number(s) of observations from which the count(s) were obtained.

Value

A numeric vector.

Author(s)

Chris Brien

See Also

angular, powerTransform.

Examples

```
n <-25
y <- rbinom(10, n, 0.5)
y <- c(y,0,n)
p.ang.mod <- angular.mod(y, n)
```

`as.alldiffs` *Forms an* `alldiffs.object` *from the supplied predictions, along with those statistics, associated with the predictions and their pairwise differences, that have been supplied.*

Description

Creates an `alldiffs.object` that consists of a `list` containing the following components: `predictions`, `vcov`, `differences`, `p.differences`, `sed`, `LSD` and `backtransforms`. Predictions must be supplied to the function while the others will be set only if they are supplied; those not supplied are set to `NULL`. It also has attributes `response`, `response.title`, `term`, `classify`, `tdf`, `tdf`, `alpha`, `sortFactor` and `sortOrder`. which will be set to the values supplied or `NULL` if none are supplied.

Usage

```
as.alldiffs(predictions, vcov = NULL, differences = NULL,
            p.differences = NULL, sed = NULL, LSD = NULL,
            backtransforms = NULL,
            response = NULL, response.title = NULL,
            term = NULL, classify = NULL,
            tdf = NULL, alpha = 0.05,
            sortFactor = NULL, sortOrder = NULL)
```

Arguments

`predictions`	A `predictions.frame`, being a `data.frame` beginning with the variables classifying the predictions and also containing columns named `predicted.value`, `standard.error` and `est.status`; each row contains a single predicted value. It may also contain columns for the lower and upper limits of error intervals for the predictions. Note that the names `standard.error` and `est.status` have been changed to `std.error` and `status` in the `pvals` component produced by `asreml-R4`; if the new names are in the `data.frame` supplied to `predictions`, they will be returned to the previous names.
`differences`	A `matrix` containing all pairwise differences between the predictions; it should have the same number of rows and columns as there are rows in `predictions`.
`p.differences`	A `matrix` containing p-values for all pairwise differences between the predictions; each p-value is computed as the probability of a t-statistic as large as or larger than the observed difference divided by its standard error. The degrees of freedom of the t distribution for computing it are computed as the denominator degrees of freedom of the F value for the fixed term, if available; otherwise, the degrees of freedom stored in the attribute `tdf` are used; the matrix should be of the same size as that for `differences`.

sed	A matrix containing the standard errors of all pairwise differences between the predictions; they are used in computing the p-values.
vcov	A matrix containing the variance matrix of the predictions; it is used in computing the variance of linear transformations of the predictions.
LSD	An LSD.frame containing the mean, minimum and maximum LSD for determining the significance of pairwise differences, as well as an assigned LSD and a measure of the accuracy of the LSD. If LSD is NULL then the LSD.frame stored in the LSD component will be calculated and the values of LSDtype, LSDby and LSDstatistic added as attributes of the alldiffs.object. The LSD for a single prediction assumes that any predictions to be compared are independent; this is not the case if residual errors are correlated.
backtransforms	A data.frame containing the backtransformed values of the predicted values that is consistent with the predictions component, except that the column named predicted.value is replaced by one called backtransformed.predictions. Any error.interval values will also be the backtransformed values. Each row contains a single predicted value.
response	A character specifying the response variable for the predictions. It is stored as an attribute to the alldiffs.object.
response.title	A character specifying the title for the response variable for the predictions. It is stored as an attribute to the alldiffs.object.
term	A character string giving the variables that define the term that was fitted using asreml and that corresponds to classify. It only needs to be specified when it is different to classify; it is stored as an attribute of the alldiffs.object. It is likely to be needed when the fitted model includes terms that involve both a numeric covariate and a factor that parallel each other; the classify would include the covariate and the term would include the factor.
classify	A character string giving the variables that define the margins of the multiway table used in the prediction. Multiway tables are specified by forming an interaction type term from the classifying variables, that is, separating the variable names with the : operator. It is stored as an attribute to the alldiffs.object.
tdf	an integer specifying the degrees of freedom of the standard error. It is used as the degrees of freedom for the t-distribution on which p-values and confidence intervals are based. It is stored as an attribute to the alldiffs.object.
alpha	A numeric giving the significance level for LSDs or one minus the confidence level for confidence intervals. It is stored as an attribute to the alldiffs.object.
sortFactor	A character containing the name of the factor that indexes the set of predicted values that determined the sorting of the components.
sortOrder	A character vector that is the same length as the number of levels for sortFactor in the predictions component of the alldiffs.object. It specifies the order of the levels in the reordered components of the alldiffs.object. The following creates a sortOrder vector levs for factor f based on the values in x: levs <- levels(f)[order(x)].

Value

An S3-class alldiffs.object. Also, see predictPlus.asreml for more information.

Author(s)

Chris Brien

See Also

asremlPlus-package, alldiffs.object, is.alldiffs, as.alldiffs, print.alldiffs, sort.alldiffs, subset.alldiffs, allDifferences.data.frame, renewClassify.alldiffs, redoErrorIntervals.alldiffs, recalcLSD.alldiffs, predictPlus.asreml, plotPredictions.data.frame, predictPresent.asreml

Examples

```
  data(Oats.dat)

  ## Use asreml to get predictions and associated statistics

  ## Not run:
  m1.asr <- asreml(Yield ~ Nitrogen*Variety,
                   random=~Blocks/Wplots,
                   data=Oats.dat)
  current.asrt <- as.asrtests(m1.asr)
  Var.pred <- asreml::predict.asreml(m1.asr, classify="Nitrogen:Variety",
                                     sed=TRUE)
  if (getASRemlVersionLoaded(nchar = 1) == "3")
    Var.pred <- Var.pred$predictions
  Var.preds <- Var.pred$pvals
  Var.sed <- Var.pred$sed
  Var.vcov <- NULL

## End(Not run)

  ## Use lmerTest and emmmeans to get predictions and associated statistics
  if (requireNamespace("lmerTest", quietly = TRUE) &
      requireNamespace("emmeans", quietly = TRUE))
  {
    m1.lmer <- lmerTest::lmer(Yield ~ Nitrogen*Variety + (1|Blocks/Wplots),
                              data=Oats.dat)
    Var.emm <- emmeans::emmeans(m1.lmer, specs = ~ Nitrogen:Variety)
    Var.preds <- summary(Var.emm)
    den.df <- min(Var.preds$df)
    ## Modify Var.preds to be compatible with a predictions.frame
    Var.preds <- as.predictions.frame(Var.preds, predictions = "emmean",
                                      se = "SE", interval.type = "CI",
                                      interval.names = c("lower.CL", "upper.CL"))
    Var.vcov <- vcov(Var.emm)
    Var.sed <- NULL
  }

  ## Use the predictions obtained with either asreml or lmerTest
  if (exists("Var.preds"))
  {
    ## Form an all.diffs object
```

```
Var.diffs <- as.alldiffs(predictions = Var.preds, classify = "Nitrogen:Variety",
                         sed = Var.sed, vcov = Var.vcov, tdf = den.df)

## Check the class and validity of the alldiffs object
is.alldiffs(Var.diffs)
validAlldiffs(Var.diffs)
}
```

`as.asrtests` *Forms an asrtests object that stores (i) a fitted asreml object, (ii) a pseudo-anova table for the fixed terms and (iii) a history of changes and hypothesis testing used in obtaining the model.*

Description

An `asrtests.object` that is a `list` consisting of the components `asreml.obj`, `wald.tab` and `test.summary`.

A call to `as.asrtests` with `test.summary = NULL` re-initializes the `test.summary` `data.frame`.

If there is no `wald.tab`, `wald.asreml` is called. In all cases, `recalcWaldTab` is called and any changes made as specified by the `recalcWaldTab` arguments supplied via `...`.

The `label` argument can be used to include an entry in `test.summary` for the starting model. If a `label` is included, (i) the information criteria calculated using the `asreml.obj` will be added to the `test.summary`, if `IClikelihood` is not set to `none` and (ii) the number of variance parameters is included in the denDF column, if `IClikelihood` is set to `none`.

Usage

```
as.asrtests(asreml.obj, wald.tab = NULL, test.summary = NULL,
            denDF = "numeric", label = NULL,
            IClikelihood = "none", bound.exclusions = c("F","B","S","C"), ...)
```

Arguments

`asreml.obj`	an `asreml` object for a fitted model.
`wald.tab`	A `data.frame` containing a pseudo-anova table for the fixed terms produced by `wald.asreml`; it should have 4 or 6 columns. Sometimes `wald.asreml` returns a `data.frame` and at other times a `list`. For example, it may return a `list` when `denDF` is used. In this case, the `Wald` component of the `list` is to be extracted and stored. It is noted that, as of asreml version 4, `wald.asreml` has a `kenadj` argument.
`test.summary`	A `data.frame` with columns `term`, `DF`, `denDF`, `p` and `action` containing the results of previous hypothesis tests.

denDF — Specifies the method to use in computing approximate denominator degrees of freedom when wald.asreml is called. Can be none to suppress the computations, numeric for numerical methods, algebraic for algebraic methods or default, the default, to automatically choose numeric or algebraic computations depending on problem size. The denominator degrees of freedom are calculated according to Kenward and Roger (1997) for fixed terms in the dense part of the model.

label — A character to use as an entry in the terms column in test.summary to indicate as far as is possible the nature of the model that has been fitted. The action column in test.summary will be Starting model.

IClikelihood — A character that controls both the occurrence and the type of likelihood for information criterion in the test.summary of the new asrtests.object. If none, none are included. Otherwise, if REML, then the AIC and BIC based on the Restricted Maximum Likelihood are included; if full, then the AIC and BIC based on the full likelihood, evaluated using REML estimates, are included. (See also infoCriteria.asreml.)

bound.exclusions — A character specifying the bound (constraint) codes that will result in a variance parameter being excluded from the count of estimated variance parameters in calculating information criteria. If set to NULL then none will be excluded.

... — further arguments passed to wald.asreml and recalcWaldTab.

Value

An object of S3-class asrtests that also inherits S3-class list.

Author(s)

Chris Brien

References

Kenward, M. G., & Roger, J. H. (1997). Small sample inference for fixed effects from restricted maximum likelihood. *Biometrics*, **53**, 983-997.

See Also

asremlPlus-package, is.alldiffs, as.alldiffs, recalcWaldTab, testranfix.asrtests, chooseModel.asrtests, rmboundary.asrtests, reparamSigDevn.asrtests

Examples

```
## Not run:
data(Wheat.dat)

# Fit initial model
current.asr <- asreml(yield ~ Rep + WithinColPairs + Variety,
                      random = ~ Row + Column + units,
```

```
                    residual = ~ ar1(Row):ar1(Column),
                    data=Wheat.dat)

# Load current fit into an asrtests object
current.asrt <- as.asrtests(current.asr, NULL, NULL)

# Check for and remove any boundary terms
current.asrt <- rmboundary(current.asrt)

## End(Not run)
```

`as.predictions.frame` *Forms a* `predictions.frame` *from a* `data.frame`*, ensuring that the correct columns are present.*

Description

Creates a `predictions.frame` from a `data.frame` by adding the `class` `predictions.frame` to it, and renaming the columns containing the `predictions`, `se`, `est.status` and `error.intervals`.

Usage

```
as.predictions.frame(data, classify = NULL,
                     predictions = NULL, se = NULL, est.status = NULL,
                     interval.type = NULL, interval.names = NULL)
```

Arguments

`data`	A `data.frame` containing columns giving the variables that uniquely index the predicted values and columns with the predicted values, their standard errors and, optionally, their estimation status (`est.status`).
`classify`	A `character` string giving the variables that define the margins of the multiway table that was predicted. Multiway tables are specified by forming an interaction type term from the classifying variables, that is, separating the variable names with the `:` operator. For predicting the overall mean, the `classify` is set to `"(Intercept)"`.
`predictions`	A `character` giving the name of the column in `data` that contains the predicted values. This column will be renamed to `predicted.value`.
`se`	A `character` giving the name of the column in `data` that contains the standard errors of the predicted values. This column will be renamed to `standard.error`.
`est.status`	A `character` giving the name of the column in `data` that contains the estimation status of the predicted values. It will have a value `Estimable` for predicted values that have been estimated and a value `Aliased` for predicted values that are `NA`. If a column named `est.status` is not present in `data` and `est.status` is `NULL`, a column `est.status` will be generated.

`interval.type` A character specifying the type of `error.intervals` stored in `data` that require renaming. If `NULL`, `error.intervals` will not be renamed, even if they are present. Otherwise, `interval.type` should be set to one of "CI", "SE" or "halfLSD".

`interval.names` A character specifying the column names of the `lower` and `upper` limits stored in `data` that are to be renamed. The `character` must be of length two, with the first element being the name of the 'lower' limit and the second element being the name of the 'upper' limit.

Value

An S3-class `predictions.frame`.

Author(s)

Chris Brien

See Also

`asremlPlus-package`, `predictions.frame`, `is.predictions.frame`, `predictions.frame`, `validPredictionsFrame`

Examples

```
  data(Oats.dat)

  ## Use asreml to get predictions and associated statistics

  ## Not run:
  m1.asr <- asreml(Yield ~ Nitrogen*Variety,
                   random=~Blocks/Wplots,
                   data=Oats.dat)
  current.asrt <- as.asrtests(m1.asr)
  Var.pred <- asreml::predict.asreml(m1.asr, classify="Nitrogen:Variety",
                                     sed=TRUE)
  if (getASRemlVersionLoaded(nchar = 1) == "3")
    Var.pred <- Var.pred$predictions
 #Form predictions.frame changing asreml-R4 names to the standard names, if these are present
 Var.preds <- as.predictions.frame(Var.pred$pvals, se = "std.error",
                                   est.status = "status")

## End(Not run)

  ## Use lmerTest and emmmeans to get predictions and associated statistics
  if (requireNamespace("lmerTest", quietly = TRUE) &
      requireNamespace("emmeans", quietly = TRUE))
  {
    m1.lmer <- lmerTest::lmer(Yield ~ Nitrogen*Variety + (1|Blocks/Wplots),
                              data=Oats.dat)
    Var.emm <- emmeans::emmeans(m1.lmer, specs = ~ Nitrogen:Variety)
    Var.preds <- summary(Var.emm)
```

```
  Var.preds <- as.predictions.frame(Var.preds, predictions = "emmean",
                                    se = "SE", interval.type = "CI",
                                    interval.names = c("lower.CL", "upper.CL"))
}

## Check the class and validity of the alldiffs object
if (exists("Var.preds"))
{
  is.predictions.frame(Var.preds)
  validPredictionsFrame(Var.preds)
}
```

---

asremlPlus-deprecated *Deprecated Functions in the Package asremlPlus*

Description

These functions have been renamed and deprecated in asremlPlus:

1. addrm.terms.asreml and addrm.terms.asrtests -> changeTerms.asrtests,
2. alldiffs -> as.alldiffs,
3. asrtests-> as.asrtests,
4. choose.model.asreml and choose.model.asrtests -> chooseModel.asrtests,
5. facRecode and facRecode.alldiffs -> facRecast.alldiffs,
6. info.crit and info.crit.asreml -> infoCriteria.asreml,
7. newrcov.asrtests -> changeTerms.asrtests,
8. plotvariofaces.asreml -> plotVariofaces.data.frame,
9. power.transform -> powerTransform,
10. predictiondiffs.asreml -> allDifferences.data.frame,
11. predictionplot.asreml -> plotPredictions.data.frame,
12. predictparallel.asreml -> predictPlus.asreml,
13. pred.present.asreml -> predictPresent.asreml,
14. recalc.wald.tab.asreml and recalc.wald.tab.asrtests -> recalcWaldTab.asrtests,
15. reorderClassify and reorderClassify.alldiffs -> renewClassify.alldiffs,
16. reml.lrt and reml.lrt.asreml -> REMLRT.asreml,
17. rmboundary.asreml -> rmboundary.asrtests,
18. setvarianceterms.asreml -> setvarianceterms.call,
19. sig.devn.reparam.asreml and sig.devn.reparam.asrtests -> reparamSigDevn.asrtests,
20. testranfix.asreml -> testranfix.asrtests,
21. testrcov.asreml and testrcov.asrtests -> testresidual.asrtests,
22. testswapran.asreml -> testswapran.asrtests

Usage

```
addrm.terms.asreml(...)
addrm.terms.asrtests(...)
alldiffs(...)
asrtests(...)
choose.model.asreml(...)
choose.model.asrtests(...)
facRecode(...)
facRecode.alldiffs(...)
info.crit(...)
info.crit.asreml(...)
newrcov.asrtests(...)
plotvariofaces.asreml(...)
power.transform(...)
predictiondiffs.asreml(...)
predictionplot.asreml(...)
predictparallel.asreml(...)
pred.present.asreml(...)
recalc.wald.tab.asreml(...)
recalc.wald.tab.asrtests(...)
reml.lrt(...)
reml.lrt.asreml(...)
## S3 method for class 'alldiffs'
reorderClassify(...)
## S3 method for class 'asreml'
rmboundary(...)
setvarianceterms.asreml(...)
sig.devn.reparam.asreml(...)
sig.devn.reparam.asrtests(...)
testranfix.asreml(...)
testrcov.asreml(...)
testrcov.asrtests(...)
## S3 method for class 'asreml'
testswapran(...)
```

Arguments

...	absorbs arguments passed from the old functions of the style foo.bar().

Author(s)

Chris Brien

`asremlPlusTips` *The randomly-presented, startup tips.*

Description

The intermittent, randomly-presented, startup tips.

Startup tips

Need help? The manual is a vignette and is in the vignettes subdirectory of the package's install directory.

Find out what has changed in asremlPlus: enter news(package = 'asremlPlus').

Need help getting started? Enter vignette(package = 'asremlPlus').

To avoid start-up message that ASReml-R is needed, load asreml before asremlPlus.

The methods for alldiffs and data.frame do not require asreml

Use suppressPackageStartupMessages() to eliminate all package startup messages.

To see all the intermittent, randomly-presented, startup tips enter ?asremlPlusTips.

To install the latest version: go to http://chris.brien.name/rpackages.

For versions between CRAN releases (and more) go to http://chris.brien.name/rpackages.

Author(s)

Chris Brien

`asrtests.object` *Description of an asrtests object*

Description

An object of S3-class `asrtests` that contains information derived from the fits of a mixed model using `asreml`.

`as.asrtests` is function that makes an object of this class.

`is.list` is the membership function for this class; it tests that an object is of class `list`.

`validAsrtests` can be used to test the validity of an `asrtests.object`.

Value

A `list` that contains three components:

1. `asreml.obj`: an object of class `asreml` that contains the fit of a model;
2. `wald.tab`: A `data.frame` containing a pseudo-anova table for the fixed terms produced by `wald.asreml`. It has `rownames` that correspond to the fixed terms that were fitted and four columns. If denominator degrees of freedom were calculated then the columns are `DF`, `denDF`, `F.inc`, `Pr`; otherwise the columns are `Df`, `Sum of Sq`, `Wald statistic`, and `Pr(Chisq)`.

3. test.summary: A data.frame with columns terms, DF, denDF, p, AIC, BIC and action, each row of which summarizes the results of proposed changes to the fitted model.

   Possible codes for action are: Dropped, Retained, Swapped, Unswapped, Unchanged, Significant, Nonsignificant, Absent, Added, Removed and Boundary. If the either of the models did not converge, unconverged will be added to the code. Unchanged is used when allow.unconverged is FALSE. Note that the logical asreml.obj$converge also reflects whether there is convergence.

   A row is added to the test.summary for each term that is dropped, added or tested or a note that several terms have been added or removed. When values for the AIC and BIC are included in the row, then the DF are the number of fixed parameters in the model and denDF are the numbers of variance parameters. When changeModelOnIC adds a row then the values of the degrees of freedom and information criteria are differences between those for the model that is supplied and the model changed by changeModelOnIC.

Author(s)

Chris Brien

See Also

as.asrtests, as.asrtests, validAsrtests

---

bootREMLRT.asreml *Uses the parametric bootstrap to calculate the p-value for a REML ratio test to compare two models.*

Description

Extracts the REML log likelihood for two asreml objects and forms the observed REML ratio statistic. It assumes that the second asreml object is the result of fitting a model that is a reduced version of the model for the first object and is considered to the null model. Using the mean and V, nboot bootstrap samples of simulated response values are generated in parallel; that is, ncores cores are used and each is used to generate and analyse a sample. The full and reduced models are fitted to the data and if either analysis fails to converge another sample is generated and analysed using the current core, with a maximum of max.retries attempts to obtain a sample that converges for both analysis. Thus the maximum number of data sets that will be generated is nboot * max.retries. If a bootstrap sample converges for both analyses, the REML ratio test statistic is formed for it. The p-value is then calculated as $(k+1)/(b+1)$ where k is the number of simulated ratio test statistics greater than the observed test statistic and s is the number of bootstrap samples that were returned.

The function checks that the models do not differ in either their fixed or sparse models. It also check the difference in the number of variance parameters between the two fits to the models, taking into account the bound.exclusions.

Usage

```
## S3 method for class 'asreml'
bootREMLRT(h0.asreml.obj, h1.asreml.obj,
           nboot = 100, max.retries = 5, seed = NULL,
           means=NULL, V = NULL, extra.matrix = NULL, ignore.terms = NULL,
           fixed.spline.terms = NULL,
           bound.exclusions = c("F","B","S","C"),
           tolerance = 1E-10, update = TRUE, trace = FALSE,
           ncores = 2, ...)
```

Arguments

h0.asreml.obj	asreml object containing the fit under the model for the null hypothesis.
h1.asreml.obj	asreml object containing the fit under the model for the alternative hypothesis.
nboot	The number of bootstrap samples to be generated.
max.retries	The maximum number of attempts to generate a sample whose analyses converge for both models.
seed	A single value, interpreted as an integer, that specifies the starting value of the random number generator. The "L'Ecuyer-CMRG" random generator is used and nextRNGStream is used to seed each core from the original seed.
means	The vector of means to be used in generating simulated bootstrap samples. If it is NULL, the fitted values based on object are used. It must be the same length as the response variable for object.
V	The fitted variance matrix, i.e. having the pattern and values that conform to the model fit stored in the supplied object. If it is NULL, estimateV.asreml is used to estimate the variance matrix for the observations from the variance parameter estimates from the reduced.asreml.obj.
extra.matrix	A matrix of order equal to the number of observations that is to be added to the variance matrix, the latter based on the information in asreml.obj. It is assumed that the sigma-parameterized values of the variance parameter estimates, such as is given in the varcomp component of summary.asreml, have been used in calculating extra.matrix; the values in the vparameters component of G.param and R.param may be either gamma- or sigma-parameterized. The argument extra.matrix can be used in conjunction with ignore.terms as a workaround to include components of the variance matrix for variance functions that have not been implemented in estimateV.
ignore.terms	A character giving terms from either the random or residual models that are to be ignored in that their contributions to the variance is not to be included in the estimated matrix. The term names are those given in the vparameters component of the asreml object or the varcomp component produced by summary.asreml, but only up to the first exclamation mark (!). This can be used in conjunction with estimateV.asreml as a workaround to include components of the variance matrix for variance functions that have not been implemented in estimateV.
fixed.spline.terms	A character vector giving one or more spline terms in the random model that are regarded as fixed and so are to be ignored because they are not regarded as contributing to the variance. The term names are those given in the

vparameters component of the asreml object or the varcomp component produced by summary.asreml, but only up to the first exclamation mark (!).

bound.exclusions
: A character specifying one or more bound codes that will result in a variance parameter in the random model being excluded from contributing to the variance. If set to NULL then none will be excluded.

tolerance
: The value such that eigenvalues less than it are considered to be zero.

update
: If TRUE then the arguments R.param and G.param are set to those in the asreml object supplied in object so that the values from the original model are used as starting values. If FALSE then asreml calls are evaluated, the only changes from the previous call being that (i) the model is fitted to simulated data and (ii) modifications specified via ... are mode, except that changes cannot be made to any of the models.

trace
: If TRUE then partial iteration details are displayed when ASReml-R functions are invoked; if FALSE then no output is displayed.

ncores
: A numeric specifying the number of cores to use in doing the simulations. In choosing a value for ncores, it is necessary to take into account other processes that are using parallel processing at the same time.

...
: Other arguments that are passed down to the function asreml. Changes to the models are not allowed. Other changes are dangerous and generally should be avoided.

Value

A list with the following components:

1. **REMLRT:** the observed REML ratio statistic.
2. **p:** the bootstrap p-value for the observed test statistic.
3. **DF:** the calculated difference in DF for the variance parameters in the two models.
4. **totalunconverged:** the total number of unconverged analyses over the simulations.
5. **REMLRT.sim:** a numeric containing the values of the ratio statistics for the simulated data. It has an attribute called na.action that can be retrieved using attr(REMLRT.sim, which = "na.action"); it contains a list of the simulation numbers that were abandoned because max.retries failed to converge for both models.
6. **nunconverged:** the number of unconverged analyses for each bootstrap sample, the maximum being max.retries.

Note

A bootstrap sample is generated using a multivariate normal distribution with expected value as specified by means and variance matrix given by V. Each simulated sample is analysed according to the reduced model and, provided this analysis converges, according to the full.model. If one of these analyses fails to converge, it is abandoned and another sample is generated for this simulation. As many as max.retries attempts are made to generate a data set for which both analyses converge. If data set that converges for both analyses is not generated for a simulation, NA is returned for that bootstrap sample. Hence, the maximum number of data sets that will be generated is nboot *

max.retries and less than nboot samples will be generated if a data set that converges for both analyses is not obtained within max.retries attempts.

If a bootstrap sample converges for both analyses, the REML ratio test statistic is calculated as $2(log(REML)_F - log(REML)_R)$.

The DF is calculated from the information in full.asreml.obj and reduced.asreml.obj. The degrees of freedom are computed as the difference between the two models in the number of variance parameters whose estimates do not have a code for bound specified in bound.exclusions.

If ASReml-R version 4 is being used then the codes specified in bound.exclusions are not restricted to a subset of the default codes, but a warning is issued if a code other than these is specified. For ASReml-R version 3, only a subset of the default codes are allowed: F (Fixed), B (Boundary), C (Constrained) and S (Singular).

Author(s)

Chris Brien

See Also

REMLRT.asreml, infoCriteria.asreml, newfit.asreml, testranfix.asrtests

Examples

```
## Not run:
    bootREMLRT(ICV.max, ICV.red, ncores = parallel::detectCores())

## End(Not run)
```

---

changeModelOnIC.asrtests

Uses information criteria to decide whether to change an already fitted model.

Description

Uses information criteria to decide whether to change the fitted model stored in the supplied asrtests.object according to the specified modifications. The function changeTerms is used to change the model. Thus, the model can be modified using a combination of adding and removing sets of terms from one or both of the fixed or random models, replacing the residual model and changing the bounds and/or initial values of some terms. The model will be unchanged if terms specified in dropFixed or dropRandom are not in the fitted model.

A row is added to the test.summary data.frame of the asrtests.object using the supplied label and stating whether or not the new model has been swapped for the supplied model. Convergence in fitting the model is checked and a note included in the action if there was not. All components of the asrtests.object are updated to exhibit the differences between the supplied and new models.

To obtain a list of the information criteria for a set of models use changeTerms.asrtests with IClikelihood set to REML or full, or use infoCriteria.asreml.

Usage

```
## S3 method for class 'asrtests'
changeModelOnIC(asrtests.obj,
                dropFixed = NULL, addFixed = NULL,
                dropRandom = NULL,  addRandom = NULL,
                newResidual = NULL,
                allow.absentDropTerms = FALSE, label = "Changed terms",
                allow.unconverged = TRUE, allow.fixedcorrelation = TRUE,
                checkboundaryonly = FALSE,
                trace = FALSE, update = TRUE, denDF = "numeric",
                set.terms = NULL, ignore.suffices = TRUE,
                bounds = "P", initial.values = NA,
                which.IC = "AIC", IClikelihood = "REML",
                fixedDF = NULL, varDF = NULL,
                bound.exclusions = c("F","B","S","C"),
           ...)
```

Arguments

asrtests.obj	An asrtests.object containing the components (i) asreml.obj, (ii) wald.tab, and (iii) test.summary.
dropFixed	A single character string in the form of a formula which, after addition of ". ~ . -" and after expansion, specifies the sum of a set of terms to be dropped from the fixed formula. The names must match those in the wald.tab component of the asrtests.obj. The fixed terms will be reordered so that single-variable terms come first, followed by two-variable terms and so on. Note that multiple terms specified using a single asreml::at function can only be dropped as a whole. If the term was specified using an asreml::at function with a single level, then it can be removed and either the level itself or its numeric position in the levels returned by the levels function can be specified.
addFixed	A single character string in the form of a formula which, after addition of ". ~ . +" and expansion, specifies the sum of a set of terms to be added to the fixed formula. The fixed terms will be reordered so that single-variable terms come first, followed by two-variable terms and so on.
dropRandom	A single character string in the form of a formula which, after addition of " ~ . -" and expansion, specifies the sum of a set of terms to be dropped from the random formula. The names must match those in the vparameters component of the asreml.obj component in the asrtests.obj. Note that multiple terms specified using a single asreml::at function can only be dropped as a whole. If the term was specified using an asreml::at function with a single level, then it can be removed and either the level itself or its numeric position in the levels returned by the levels function can be specified.
addRandom	A single character string in the form of a formula which, after addition of " ~ . +"and expansion, specifies the sum of a set of terms to be added to the random formula.
newResidual	A single character string in the form of a formula which, after addition of " ~ ", specifies the residual (or rcov) model. To remove the model, enter "-(.)".

allow.absentDropTerms
: A logical indicating whether to to change the model when terms specified in dropFixed or dropRandom are not in the fitted model.

label
: A character to use as an entry in the terms column in test.summary to indicate as far as is possible the terms that are being manipulated.

allow.unconverged
: A logical indicating whether to accept a new model even when it does not converge. If FALSE and the fit of the new model does not converge, the supplied asrtests.obj is returned. Also, if FALSE and the fit of the new model has converged, but that of the old model has not, the new model will be accepted.

allow.fixedcorrelation
: A logical indicating whether to accept a new model even when it contains correlations in the model whose values have been designated as fixed, bound or singular. If FALSE and the new model contains correlations whose values have not been able to be estimated, the supplied asrtests.obj is returned. The fit in the asreml.obj component of the supplied asrtests.obj will also be tested and a warning issued if both fixed correlations are found in it and allow.fixedcorrelation is FALSE.

checkboundaryonly
: If TRUE then boundary and singular terms are not removed by rmboundary.asrtests; a warning is issued instead.

trace
: If TRUE then partial iteration details are displayed when ASReml-R functions are invoked; if FALSE then no output is displayed.

update
: If TRUE, and set.terms is NULL, then newfit.asreml is called to fit the model to be tested, using the values of the variance parameters stored in the asreml.object, that is stored in asrtests.obj, as starting values. If FALSE or set.terms is not NULL, then newfit.asreml will not use the stored variance parameter values as starting values when fitting the new model, the only modifications being (i) those specified by this function's arguments and (ii) those specified via ....

denDF
: Specifies the method to use in computing approximate denominator degrees of freedom when wald.asreml is called. Can be none to suppress the computations, numeric for numerical methods, algebraic for algebraic methods or default, the default, to automatically choose numeric or algebraic computations depending on problem size. The denominator degrees of freedom are calculated according to Kenward and Roger (1997) for fixed terms in the dense part of the model.

set.terms
: A character vector specifying the terms that are to have bounds and/or initial values set prior to fitting the new model. The names must match those in the vparameters component of the asreml.obj component in the new asrtests.object. The terms in the model do not need to change from those in the model in the supplied asrtests.obj.

ignore.suffices
: A logical vector specifying whether the suffices of the asreml-assigned names of the variance terms (i.e. the information to the right of an "!", other than "R!") is to be ignored in matching elements of terms. If TRUE for an element of terms, the suffices are stripped from the asreml-assigned names. If FALSE for an element of terms, the element must exactly match an asreml-assigned

name for a variance term. This vector must be of length one or the same length as terms. If it is of length one then the same action is applied to the asreml-assigned suffices for all the terms in terms.

bounds A character vector specifying the bounds to be applied to the terms specified in set.terms. This vector must be of length one or the same length as set.terms. If it is of length one then the same constraint is applied to all the terms in set.terms. If any of the bounds are equal to NA then they are left unchanged for those terms.

which.IC A character specifying the information criterion to be used in selecting the best model. Possible values are AIC and BIC. The value of the criterion for supplied model must exceed that for changed model for the changed model to be returned.

IClikelihood A character specifying whether Restricted Maximum Likelihood (REML) or the full likelihood, evaluated using REML estimates, (full) are to be used in calculating the information criteria for choosing between models when family is set to asr_gaussian. For family set to asr_binomial or asr_poisson and with dispersion set to 1, the deviance is extracted from object and used to calculate the AIC and BIC (as suggested by Damian Collins); the setting of IClikelihood is ignored and the log-likelihood set to NA. The information criteria are not valid for other settings of family and dispersion.

fixedDF A numeric giving the number of estimated fixed parameters. If NULL then this is determined from the information in asreml.obj.

varDF A numeric giving the number of estimated variance parameters. If NULL then this is determined from the information in asreml.obj. It replaces the varDF argument.

initial.values A character vector specifying the initial values for the terms specified in terms. This vector must be of length one or the same length as terms. If it is of length one then the same initial value is applied to all the terms in terms. If any of the initial.values are equal to NA then they are left unchanged for those terms.

bound.exclusions
A character specifying the bound (constraint) codes that will result in a variance parameter being excluded from the count of estimated variance parameters in calculating the information criteria. If set to NULL then none will be excluded.

... Further arguments passed to asreml, wald.asreml and as.asrtests.

Value

An asrtests.object containing the components (i) asreml.obj, (ii) wald.tab, and (iii) test.summary. The values of the degrees of freedom and the information criteria are differences between those of the changed model and those of the model supplied to changeModelOnIC.

Author(s)

Chris Brien

See Also

as.asrtests, rmboundary.asrtests, testranfix.asrtests, testresidual.asrtests, newfit.asreml, reparamSigDevn.asrtests, chooseModel.asrtests, changeTerms.asrtests, infoCriteria.asreml

Examples

```
## Not run:

data(Wheat.dat)
current.asr <- asreml(yield ~ Rep + WithinColPairs + Variety,
                      random = ~ Row + Column + units,
                      residual = ~ ar1(Row):ar1(Column),
                      data=Wheat.dat)
current.asrt <- as.asrtests(current.asr, NULL, NULL,
                            label = "Maximal model")
current.asrt <- rmboundary(current.asrt)

# Drop both Row and Column terms
current.asrt <- changeModelOnIC(current.asrt,
                                dropRandom = "Row + Column",
                                checkboundaryonly = TRUE,
                                which.IC = "AIC", IClikelihood = "full")
current.asrt <- iterate(current.asrt)

# Add and drop both fixed and random terms
current.asrt <- changeModelOnIC(current.asrt,
                                addFixed = "vRow", dropFixed = "WithinColPairs",
                                addRandom = "spl(vRow)", dropRandom = "units",
                                checkboundaryonly = TRUE,
                                which.IC = "AIC", IClikelihood = "full")

# Replace residual with model without Row autocorrelation
current.asrt <- changeModelOnIC(current.asrt,
                                newResidual = "Row:ar1(Column)",
                                label="Row autocorrelation",
                                IClikelihood = "full")


## End(Not run)
```

---

changeTerms.asrtests *Adds and drops terms from one or both of the fixed or random model, replaces the residual (rcov) model with a new model and changes bounds or initial values of terms.*

Description

The specified terms are simply added or dropped, without testing, from either the fixed or random model and/or the residual (rcov) model replaced. Also, the bounds and/or initial values of some

terms can be changed. No hypothesis testing is performed, but a check is made for boundary or singular terms.

A row is added to the test.summary data.frame of the asrtests.object using the supplied label and stating which models have been changed. Information criteria can be included in the row of the test.summary. Convergence in fitting the model is checked and a note included in the action if there was not. All components of the asrtests.object are updated.

To only change the terms based on a comparison of information criteria use changeModelOnIC.asrtests.

Usage

```
## S3 method for class 'asrtests'
changeTerms(asrtests.obj,
            dropFixed = NULL, addFixed = NULL,
            dropRandom = NULL,  addRandom = NULL,
            newResidual = NULL, label = "Changed terms",
            allow.unconverged = TRUE, allow.fixedcorrelation = TRUE,
            checkboundaryonly = FALSE,
            trace = FALSE, update = TRUE, denDF = "numeric",
            set.terms = NULL, ignore.suffices = TRUE,
            bounds = "P", initial.values = NA,
            IClikelihood = "none", bound.exclusions = c("F","B","S","C"),
            ...)
```

Arguments

asrtests.obj	An asrtests.object containing the components (i) asreml.obj, (ii) wald.tab, and (iii) test.summary.
dropFixed	A single character string in the form of a formula which, after addition of ". ~ . -" and after expansion, specifies the sum of a set of terms to be dropped from the fixed formula. The names must match those in the wald.tab component of the asrtests.obj. The fixed terms will be reordered so that single-variable terms come first, followed by two-variable terms and so on. Note that multiple terms specified using a single asreml::at function can only be dropped as a whole. If the term was specified using an asreml::at function with a single level, then it can be removed and either the level itself or its numeric position in the levels returned by the levels function can be specified.
addFixed	A single character string in the form of a formula which, after addition of ". ~ . +" and expansion, specifies the sum of a set of terms to be added to the fixed formula. The fixed terms will be reordered so that single-variable terms come first, followed by two-variable terms and so on.
dropRandom	A single character string in the form of a formula which, after addition of " ~ . -" and expansion, specifies the sum of a set of terms to be dropped from the random formula. The terms must match those in the vparameters component of the asreml.obj component in the asrtests.obj. Note that multiple terms specified using a single asreml::at function can only be dropped as a whole. If the term was specified using an asreml::at function with a single level, then it can be removed and either the level itself or its numeric position in the levels returned by the levels function can be specified.

addRandom A single character string in the form of a formula which, after addition of " ~ . +"and expansion, specifies the sum of a set of terms to be added to the random formula.

newResidual A single character string in the form of a formula which, after addition of " ~ ", specifies the residual (or rcov) model. To remove the model, enter "-(.)".

label A character to use as an entry in the terms column in test.summary to indicate as far as is possible the terms that are being manipulated.

allow.unconverged
A logical indicating whether to accept a new model even when it does not converge. If FALSE and the fit does not converge, the supplied asrtests.obj is returned.

allow.fixedcorrelation
A logical indicating whether to accept a new model even when it contains correlations in the model whose values have been designated as fixed, bound or singular. If FALSE and the new model contains correlations whose values have not been able to be estimated, the supplied asrtests.obj is returned. The fit in the asreml.obj component of the supplied asrtests.obj will also be tested and a warning issued if both fixed correlations are found in it and allow.fixedcorrelation is FALSE.

checkboundaryonly
If TRUE then boundary and singular terms are not removed by rmboundary.asrtests; a warning is issued instead.

trace If TRUE then partial iteration details are displayed when ASReml-R functions are invoked; if FALSE then no output is displayed.

update If TRUE, and set.terms is NULL, then newfit.asreml is called to fit the model to be tested, using the values of the variance parameters stored in the asreml.object, that is stored in asrtests.obj, as starting values. If FALSE or set.terms is not NULL, then newfit.asreml will not use the stored variance parameter values as starting values when fitting the new model, the only modifications being (i) those specified by this function's arguments and (ii) those specified via ....

denDF Specifies the method to use in computing approximate denominator degrees of freedom when wald.asreml is called. Can be none to suppress the computations, numeric for numerical methods, algebraic for algebraic methods or default, the default, to automatically choose numeric or algebraic computations depending on problem size. The denominator degrees of freedom are calculated according to Kenward and Roger (1997) for fixed terms in the dense part of the model.

set.terms A character vector specifying the terms that are to have bounds and/or initial values set prior to fitting the new model. The names must match those in the vparameters component of the asreml.obj component in the new asrtests.object. The terms in the model do not need to change from those in the model in the supplied asrtests.obj.

ignore.suffices
A logical vector specifying whether the suffices of the asreml-assigned names of the variance terms (i.e. the information to the right of an "!", other than "R!") is to be ignored in matching elements of terms. If TRUE for an element

of terms, the suffices are stripped from the asreml-assigned names. If FALSE for an element of terms, the element must exactly match an asreml-assigned name for a variance term. This vector must be of length one or the same length as terms. If it is of length one then the same action is applied to the asreml-assigned suffices for all the terms in terms.

bounds — A character vector specifying the bounds to be applied to the terms specified in set.terms. This vector must be of length one or the same length as set.terms. If it is of length one then the same constraint is applied to all the terms in set.terms. If any of the bounds are equal to NA then they are left unchanged for those terms.

initial.values — A character vector specifying the initial values for the terms specified in terms. This vector must be of length one or the same length as terms. If it is of length one then the same initial value is applied to all the terms in terms. If any of the initial.values are equal to NA then they are left unchanged for those terms.

IClikelihood — A character that controls both the occurrence and the type of likelihood for information criterion in the test.summary of the new asrtests.object. If none, none are included. Otherwise, if REML and family is set to asr_guassian (the default), then the AIC and BIC based on the Restricted Maximum Likelihood are included; if full and family is set to asr_guassian, then the AIC and BIC based on the full likelihood, evaluated using REML estimates, are included. if family is asr_binomial or asr_poisson, with dispersion set to 1, the deviance is extracted from object and used to calculate the AIC and BIC. (See also infoCriteria.asreml.)

bound.exclusions — A character specifying the bound (constraint) codes that will result in a variance parameter being excluded from the count of estimated variance parameters in calculating the information criteria. If set to NULL then none will be excluded.

... — Further arguments passed to asreml, wald.asreml and as.asrtests.

Value

An asrtests.object containing the components (i) asreml.obj, (ii) wald.tab, and (iii) test.summary.

Author(s)

Chris Brien

References

Kenward, M. G., & Roger, J. H. (1997). Small sample inference for fixed effects from restricted maximum likelihood. *Biometrics*, **53**, 983-997.

See Also

as.asrtests, rmboundary.asrtests, testranfix.asrtests, testresidual.asrtests, newfit.asreml, reparamSigDevn.asrtests, chooseModel.asrtests, changeModelOnIC.asrtests, infoCriteria.asreml

Examples

```
## Not run:
terms <- "(Date/(Sources * (Type + Species)))"
current.asrt <- changeTerms(current.asrt, addFixed = terms)

current.asrt <- changeTerms(current.asrt, dropFixed = "A + B", denDF = "algebraic")

data(Wheat.dat)
current.asr <- asreml(yield ~ Rep + WithinColPairs + Variety,
                      random = ~ Row + Column + units,
                      residual = ~ ar1(Row):ar1(Column),
                      data=Wheat.dat)
current.asrt <- as.asrtests(current.asr, NULL, NULL)
current.asrt <- rmboundary(current.asrt)
# Add and drop both fixed and random terms
current.asrt <- changeTerms(current.asrt,
                            addFixed = "vRow", dropFixed = "WithinColPairs",
                            addRandom = "spl(vRow)", dropRandom = "units",
                            checkboundaryonly = TRUE)
# Replace residual with model without Row autocorrelation
current.asrt <- changeTerms(current.asrt,
                            newResidual = "Row:ar1(Column)",
                            label="Row autocorrelation")


## End(Not run)
```

ChickpeaEnd.dat *A large data set comprising the end of imaging data from a chick pea experiment conducted in high-throughput greenhouses*

Description

The data collected after imaging had been completed on the 1056 plants in the experiment reported by Atieno et al. (2017). The design employed for the experiment was a split-plot design in which two consecutive carts formed a main plot. The split-plot design assigned 245 genotypes to main plots, the genotypes being unequally replicated 2 or 3 times. Treatments (non-saline, saline) were randomized to the two subplots (carts) within each main plot.

The columns in the `data.frame` are: Smarthouse, Lane, Position, Zone, Mainplot, Subplot, Replicate, xLane, xPosition, Genotypes, Treatments, Biomass, PlantHeight, SenescenceRank, TotalPods, FilledPods, EmptyPods, SeedNo, TotalSeedWt, SeedWt100.

The columns Smarthouse, Lane and Position uniquely identify the rows of observations. Zones are groups of 4 Lanes, Mainplots are the 44 pairs of consecutive Subplots within each Zone, and a Subplot is a cart containing a single plant. The columns xLane and xPosition are numeric covariates for location within a Smarthouse. Genotypes and Treatments indicate the genotype and treatment that each plant was allocated. The response variables are Biomass, PlantHeight, SenescenceRank, TotalPods, FilledPods, EmptyPods, SeedNo, TotalSeedWt and SeedWt100.

Usage

```
data(ChickpeaEnd.dat)
```

Format

A data.frames with 1056 rows by 20 columns.

References

Atieno, J., Li, Y., Langridge, P., Dowling, K., Brien, C., Berger, B., Varshney, R. K., and Sutton, T. (2017). Exploring genetic variation for salinity tolerance in chickpea using image-based phenotyping. *Scientific Reports*, **7**, 1300. doi:10.1038/s41598017012117

---

chooseModel *Determines the set of significant terms using p-values and records the tests performed in a data.frame, taking into account the marginality relations of terms.*

Description

Using p-values from hypothesis tests, determines the set of significant terms, taking into account the hierarchy or marginality of terms. In particular, a term will not be tested if it is marginal to (or nested in) one that is significant. For example, if A:B is significant, then neither A nor B will be tested. The tests conducted in choosing selected model are listed in a summary data.frame.

Usage

```
chooseModel(object, ...)
```

Arguments

object	an object using which p-values can be obtained for use in model selection.
...	further arguments passed to or from other methods.

Details

chooseModel is the generic function for the chooseModel method. Use methods("chooseModel") to get all the methods for the chooseModel generic.

chooseModel.asrtests is a method for an asrtests.object. It uses testranfix.asrtests to conduct tests to determine the p-values used in the model selection.

chooseModel.data.frame is a method for a data.frame. It uses the p-values stored in the data.frame in the model selection.

Author(s)

Chris Brien

See Also

chooseModel.asrtests, chooseModel.asrtests, changeModelOnIC.asrtests, testranfix.asrtests

---

chooseModel.asrtests *Determines and records the set of significant terms using an* asrtests.object*, taking into account the hierarchy or marginality relations of the terms.*

Description

Performs a series of hypothesis tests on a set of fixed and/or random terms taking into account the marginality of terms. In particular, a term will not be tested if it is marginal to (or nested in) one that is significant. For example, if A:B is significant, then neither A nor B will be tested. For a random term, the term is removed from the model fit, any boundary terms are removed using rmboundary.asrtests and a REML likelihood ratio test is performed using REMLRT.asreml. If it is not significant and drop.ran.ns is TRUE, the term is permanently removed from the model. Note that if boundary terms are removed, the reduced model may not be nested in the full model in which case the test is not valid. For fixed terms, the Wald tests are performed and the p-value for the term obtained. If it is not significant and drop.fix.ns is TRUE, the term is permanently removed from the model. A row that records the outcome of a test is added to test.summary for each term that is tested.

Usage

```
## S3 method for class 'asrtests'
chooseModel(object, terms.marginality=NULL,
            alpha = 0.05, allow.unconverged = TRUE,
            allow.fixedcorrelation = TRUE,
            checkboundaryonly = FALSE, drop.ran.ns=TRUE,
            positive.zero = FALSE, bound.test.parameters = "none",
            drop.fix.ns=FALSE, denDF = "numeric",  dDF.fault = "none",
            dDF.values = NULL, trace = FALSE, update = TRUE,
            set.terms = NULL, ignore.suffices = TRUE,
            bounds = "P", initial.values = NA,
            IClikelihood = "none", ...)
```

Arguments

object	an asrtests.object containing the components (i) asreml.obj, (ii) wald.tab, and (iii) test.summary.
terms.marginality	A square matrix of ones and zeros with row and column names being the names of the terms to be tested. The names of fixed terms must match those in the wald.tab component of the object, while the names of random terms must match those in the vparameters component of the asreml.obj component in the object. The diagonal elements of the matrix should be one, indicating that a term is marginal to itself. Elements should be one if the row term is marginal to the column term. All other elements should be zero.

alpha The significance level for the test.

allow.unconverged
A logical indicating whether to accept a new model even when it does not converge. If FALSE and a fit when a term is removed does not converge, the term will not be removed.

allow.fixedcorrelation
A logical indicating whether to accept a new model even when it contains correlations in the model whose values have been designated as fixed, bound or singular. If FALSE and the new model contains correlations whose values have not been able to be estimated, the supplied asrtests.obj is returned. The fit in the asreml.obj component of the supplied asrtests.obj will also be tested and a warning issued if both fixed correlations are found in it and allow.fixedcorrelation is FALSE.

checkboundaryonly
If TRUE then boundary and singular terms are not removed by rmboundary.asrtests; a warning is issued instead.

drop.ran.ns A logical indicating whether to drop nonsignificant random terms from the model.

positive.zero Indicates whether the hypothesized values for the variance components being tested are on the boundary of the parameter space. For example, this is true for positively-constrained variance components that, under the reduced model, are zero. This argument does not need to be set if bound.test.parameters is set.

bound.test.parameters
Indicates whether for the variance components being tested, at least some of the hypothesized values are on the boundary of the parameter space. The possibilities are "none", "onlybound" and "one-and-one". The default is "none", although if it is set to "none" and positive.zero is TRUE then bound.test.parameters is taken to be "onlybound". When bound.test.parameters is set to "one-and-one", it signifies that there are two parameters being tested, one of which is bound and the other is not. For example, the latter is true for testing a covariance and a positively-constrained variance component that, under the reduced model, are zero.

drop.fix.ns A logical indicating whether to drop a fixed term from the model when it is nonsignificant

denDF Specifies the method to use in computing approximate denominator degrees of freedom when wald.asreml is called. Can be none to suppress the computations, numeric for numerical methods, algebraic for algebraic methods or default, the default, to automatically choose numeric or algebraic computations depending on problem size. The denominator degrees of freedom are calculated according to Kenward and Roger (1997) for fixed terms in the dense part of the model.

dDF.fault A character specifying the method to use to obtain substitute denominator degrees of freedom. when the numeric or algebraic methods produce faulty values, viz. NA, Inf or less than 0.01. Consistent with when no denDF are available, the default is "residual" and so the residual degrees of freedom from asreml.obj$nedf are used. If dDF.fault = "none", no substitute denominator degrees of freedom are employed; if dDF.fault = "residual"; if dDF.fault = "maximum", the maximum of those denDF that are available, excluding that

for the Intercept, is used; if all denDF are faulty, asreml.obj$nedf is used. If dDF.fault = "supplied", a vector of values for the denominator degrees of freedom is to be supplied in dDF.values. Any other setting is ignored and a warning message produced. Generally, substituting these degrees of freedom is anticonservative in that it is likely that the degrees of freedom used will be too large.

dDF.values A vector of values to be used when dDF.fault = "supplied". Its values will be used when denDF in a test for a fixed effect is NA. This vector must be the same length as the number of fixed terms, including (Intercept) whose value could be NA.

trace If TRUE then partial iteration details are displayed when ASReml-R functions are invoked; if FALSE then no output is displayed.

update If TRUE, and set.terms is NULL, then newfit.asreml is called to fit the model to be tested, using the values of the variance parameters stored in the asreml.object, that is stored in asrtests.obj, as starting values. If FALSE or set.terms is not NULL, then newfit.asreml will not use the stored variance parameter values as starting values when fitting the new model, the only modifications being (i) to the terms in the fixed and random models corresponding to terms in terms.marginality and (ii) those specified via ....

set.terms A character vector specifying the terms that are to have bounds and/or initial values set prior to fitting. The names must match those in the vparameters component of the asreml.obj component in the new asrtests.object.

ignore.suffices
A logical vector specifying whether the suffices of the asreml-assigned names of the variance terms (i.e. the information to the right of an "!", other than "R!") is to be ignored in matching elements of terms. If TRUE for an element of terms, the suffices are stripped from the asreml-assigned names. If FALSE for an element of terms, the element must exactly match an asreml-assigned name for a variance term. This vector must be of length one or the same length as terms. If it is of length one then the same action is applied to the asreml-assigned suffices for all the terms in terms.

bounds A character vector specifying the bounds to be applied to the terms specified in set.terms. This vector must be of length one or the same length as set.terms. If it is of length one then the same constraint is applied to all the terms in set.terms. If any of the bounds are equal to NA then they are left unchanged for those terms.

initial.values A character vector specifying the initial values for the terms specified in terms. This vector must be of length one or the same length as terms. If it is of length one then the same initial value is applied to all the terms in terms. If any of the initial.values are equal to NA then they are left unchanged for those terms.

IClikelihood A character that controls both the occurrence and the type of likelihood for information criterion in the test.summary of the new asrtests.object. If none, none are included. Otherwise, if REML, then the AIC and BIC based on the Restricted Maximum Likelihood are included; if full, then the AIC and BIC based on the full likelihood, evaluated using REML estimates, are included. (See also infoCriteria.asreml.)

... further arguments passed to asreml, wald.asreml and as.asrtests via testranfix.asrtests.

Value

A list containing:

1. `asrtests.obj`: an `asrtests.object` containing the components (i) `asreml.obj`, (ii) `wald.tab`, and (iii) `test.summary`.;
2. `sig.tests`: a `character vector` whose elements are the significant terms amongst those tested.

Author(s)

Chris Brien

References

Kenward, M. G., & Roger, J. H. (1997). Small sample inference for fixed effects from restricted maximum likelihood. *Biometrics*, **53**, 983-997.

See Also

`chooseModel`, `chooseModel.data.frame`, `as.asrtests`, `testranfix.asrtests`, `testresidual.asrtests`, `REMLRT.asreml`, `rmboundary.asrtests`, `newfit.asreml`, `changeModelOnIC.asrtests`, `changeTerms.asrtests`, `reparamSigDevn.asrtests`

Examples

```
## Not run:
data(WaterRunoff.dat)
asreml.options(keep.order = TRUE) #required for asreml-R4 only
current.asr <- asreml(log.Turbidity ~ Benches + (Sources * (Type + Species)) * Date,
                      random = ~Benches:MainPlots:SubPlots:spl(xDay),
                      data = WaterRunoff.dat, keep.order = TRUE)
current.asrt <- as.asrtests(current.asr, NULL, NULL)
terms.treat <- c("Sources", "Type", "Species",
                 "Sources:Type", "Sources:Species")
terms <- sapply(terms.treat,
                FUN=function(term){paste("Date:",term,sep="")},
                simplify=TRUE)
terms <- c("Date", terms)
terms <- unname(terms)
marginality <-  matrix(c(1,0,0,0,0,0, 1,1,0,0,0,0,  1,0,1,0,0,0,
                         1,0,1,1,0,0, 1,1,1,0,1,0, 1,1,1,1,1,1), nrow=6)
rownames(marginality) <- terms
colnames(marginality) <- terms
choose <- chooseModel(current.asrt, marginality)
current.asrt <- choose$asrtests.obj
sig.terms <- choose$sig.terms

## End(Not run)
```

chooseModel.data.frame

Determines the set of significant terms from results stored in a data.frame*, taking into account the marginality relations of terms and recording the tests used in a* data.frame.

Description

Uses the p.values from a set of hypothesis tests that are stored in the supplied data.frame to choose a model to describe the effects of the terms corresponding to the p-values, taking into account the hierarchy or marginality of terms. In particular, a term will not be tested if it is marginal to (or nested in) one that is significant. For example, if A:B is significant, then neither A nor B will be tested. The tests used in choosing the selected model are listed in the data.frame choose.summary.

No change is made to the p.values, the DF and denDF being for information only.

Usage

```
## S3 method for class 'data.frame'
chooseModel(object, terms=NULL, p.values = "Pr",
            DF = "Df", denDF = "denDF", omit.DF = FALSE,
            terms.marginality=NULL, alpha = 0.05, ...)
```

Arguments

object	a data.frame object containing the results of hypothesis tests for a set of terms. It components should include terms, p.values, and, if not set to NA, DF and denDF.
terms	A character giving the name of the column in object containing the terms corresponding to the p.values. If NULL, it is assumed that the row names of object give the terms.
p.values	A character giving the name of the column in object containing the p-values to use in deciding whether or not terms are significant.
DF	Can be a character or a numeric that specifies the numerator degrees of freedom for the tests. If it is a character, it must be the name of a column in object containing the numerator degrees of freedom that are to be included in the choose.summary data.frame. if it is a numeric, its length must equal 1 or the number of rows in object. In either case, a column labelled DF will be included in the choose.summary data.frame. It will contain either the replicated single value (which can be NA) or the values supplied.
denDF	Can be a character or a numeric that specifies the denominator degrees of freedom for the tests. If it is a character, it must be the name of a column in object containing the denominator degrees of freedom that are to be included in the choose.summary data.frame. if it is a numeric, its length must equal 1 or the number of rows in object. In either case, a column labelled denDF will be included in the choose.summary data.frame. It will contain either the replicated single value (which can be NA) or the values supplied.

omit.DF — A logical indicating whether or not both the numerator and denominator degrees of freedom are to be omitted from choose.summary. Doing so will mean that the choose.summary no longer has the same columns as a test.summary from an asrtests.object.

terms.marginality — A square matrix of ones and zeros with row and column names being the names of the those terms in the terms column of object that are to be tested. The diagonal elements should be one, indicating that a term is marginal to itself. Elements should be one if the row term is marginal to the column term. All other elements should be zero. The names of the rows and columns should match the those elements of terms that are to be tested.

alpha — The significance level for the hypothesis testing.

... — Provision for passing arguments to functions called internally - not used at present.

Value

A list containing:

1. choose.summary: a data.frame summarizing the tests carried out in choosing the significant terms; provided omit.DF = FALSE, it has the same columns as a test.summary from an asrtests.object
2. sig.tests: a character vector whose elements are the significant terms amongst those tested.

Author(s)

Chris Brien

See Also

chooseModel, chooseModel.asrtests

Examples

```
data("Ladybird.dat")

## Use asreml to get the table of p-values

## Not run:
m1.asr <- asreml(logitP ~ Host*Cadavers*Ladybird,
                 random = ~ Run,
                 data = Ladybird.dat)
current.asrt <- as.asrtests(m1.asr)
fixed.tab <-  current.asrt$wald.tab
col.p <- "Pr"
df = "Df"
den.df = "denDF"

## End(Not run)
```

```
## Use lmeTest to get the table of p-values
if (requireNamespace("lmerTest", quietly = TRUE) &
    requireNamespace("emmeans", quietly = TRUE))
{
  m1.lmer <- lmerTest::lmer(logitP ~ Host*Cadavers*Ladybird + (1|Run),
                            data=Ladybird.dat)
  fixed.tab <- anova(m1.lmer, type = "II")
  col.p <- "Pr(>F)"
  df = "NumDF"
  den.df = "DenDF"
}

## Select a model using the table of p-values obtained with either asreml or lmerTest
if (exists("fixed.tab"))
{
  term.marg <- dae::marginality(dae::pstructure(~ Host*Cadavers*Ladybird,
                                                data = Ladybird.dat))
  chosen <- chooseModel(fixed.tab, p.values = col.p, DF = df, denDF = den.df,
                        terms.marginality = term.marg)
}
```

`chooseSpatialModelOnIC.asrtests`
Uses information criteria to choose the best fitting spatial model for accounting for local spatial variation.

Description

For a response variable measured on a potentially irregular grid of rows and columns of the units, uses information criteria (IC) to decide whether the fit and parsimony of the model fitted to a set of data can be improved by adding, to the fitted model stored in the supplied `asrtests.object`, one of the following spatial models to account for the local spatial variation: (i) a two-dimensional first-order autocorrelation model, (ii) a two-dimensional tensor-product natural cubic smoothing spline model (TPNCSS), (iii) a two-dimensional tensor-product penalized P-spline model with second-difference penalties (TPPSC2) model, or (iv) a two-dimensional tensor-product penalized linear spline model with first-difference penalties (TPPSL1). The models from which to select can be reduced to a subset of these four models. For each model, a term from the spatial model is only added to the supplied model if the IC of the supplied model is decreased with the addition of that term. If no model improves the IC when a local spatial variation model is added, then the supplied, nonspatial model will be returned. The data can be arranged in sections, for each of which there is a grid and for which the model is to be fitted separately. Also, the rows and columns of a grid are not necessarily one observational unit wide. For `TPPSC2` models, the improvement in the fit from rotating the eigenvectors of the penalty matrix can be investigated; if there is no improvement, the unrotated fit will be returned.

One or more rows is added to the `test.summary` `data.frame` of the `asrtests.object`, for each `section` and each spatial model, stating whether or not the new model has been swapped for a model in which the spatial model has been added to the supplied model. Convergence in fitting

the model is checked and a note included in the action if there was not. All components of the asrtests.object are updated to exhibit the differences between the supplied and any new model.

Usage

```
## S3 method for class 'asrtests'
chooseSpatialModelOnIC(asrtests.obj, trySpatial = "all",
                       sections = NULL,
                       row.covar = "cRow", col.covar = "cCol",
                       row.factor = "Row", col.factor = "Col",
                       corr.funcs = c("ar1", "ar1"), corr.orders = c(0, 0),
                       row.corrFitfirst = TRUE,
                       allow.corrsJointFit = TRUE, nugget.variance = TRUE,
                       dropFixed = NULL, dropRandom = NULL,
                       nsegs = NULL, nestorder = c(1,1),
                       usRandLinCoeffs = TRUE,
                       rotateX = FALSE, ngridangles = NULL,
                       which.rotacriterion = "AIC", nrotacores = 1,
                       asreml.option = "grp", tpps4mbf.obj = NULL,
                       allow.unconverged = TRUE, allow.fixedcorrelation = TRUE,
                       checkboundaryonly = FALSE, update = TRUE, trace = FALSE,
                       maxit = 30, IClikelihood = "full", which.IC = "AIC",
                       return.asrts = "best", ...)
```

Arguments

asrtests.obj	An asrtests.object containing the components (i) asreml.obj, (ii) wald.tab, and (iii) test.summary.
trySpatial	A character string nominating the types of spatial model whose fits are to be assessed. Possible values are none, all, corr, TPNCSS, TPPSC2 (or TPPCS), and TPPSL1 (or TPP1LS). If set to none, then just the supplied nonspatial model and the information about its information criteria will be returned. If all, then corr, TPNCSS, TPPSC2 and TPPSL1 will be fitted. Which fitted models are returned is controlled by return.asrts.
sections	A single character string that specifies the name of the column in the data.frame that contains the factor that identifies different sections of the data to which separate spatial models are to be fitted. Note that, for other terms that involve sections in the random formula, there should be separate terms for each level of sections. For example, in a blocked experiment involving multiple sites, there should be the sum of separate terms for the Blocks at each Site i.e. a formula that contains terms like at(Site, i):Block for each site and these are separated by '+'. Otherwise, the combined term (e.g. Site:Block) will impact on the fitting of the local spatial models for the different Sites. Similarly, a separate residual variance for each of the sections should be fitted, unless there is a need to fit a different variance structure to the residual, e.g. heterogeneous residual variances depending on treatments. Separate residual variances for sections can be achieved using the asreml functions dsum or idh. Because, unlike random terms, terms for residual variances are not removed from the model, compound residual terms can be used to include them in the model, e.g. terms with

idh or dsum with multiple levels in the list or leaving levels out altogether. In addition to allowing the independent fitting of models to the sections, separate residual variance terms allows a nugget variance to be fitted in a correlation model for each of the sections.

row.covar A single character string nominating a numeric that contains the values of a centred covariate indexing the rows of a grid. The numeric must be a column in the data.frame stored in the asreml.obj that is a component of the supplied asrtests.obj.

col.covar A single character string nominating a numeric that contains the values of a centred covariate indexing the columns of a grid. The numeric must be a column in the data.frame stored in the asreml.obj that is a component of the supplied asrtests.obj.

row.factor A single character string nominating a factor that indexes the rows of a grid that are to be one dimension of a spatial correlation model. The factor must a column in the data.frame stored in the asreml.obj that is a component of the supplied asrtests.obj.

col.factor A single character string nominating a factor that indexes the columns of a grid that are to be one dimension of a spatial correlation model. The factor must a column in the data.frame stored in the asreml.obj that is a component of the supplied asrtests.obj.

corr.funcs A single character string of length two that specifies the asreml one-dimensional correlation or variance model function for the row and column dimensions of a two-dimensional separable spatial correlation model to be fitted when spatial.model is corr; the two-dimensional model is fitted as a random term. If a correlation or variance model is not to be investigated for one of the dimensions, specify "" for that dimension. If the correlation model is corb, the values of corr.orders are used for its order argument (b).

corr.orders A numeric of length two that specifies the order argument (b) values for the row and column dimensions of a two-dimensional separable spatial correlation model when spatial.model is corr and the corr.funcs for a dimension is corb, the asreml banded correlation model. If one of the dimensions does not involve an order argument, set the value of corr.orders for that dimension to zero. For a dimension for which the corr.funcs is corb and corr.orders is zero, a model with a single band, the correlation between immediate neighbours, will be fitted and then further bands, up to a maximum of 10 bands, will be added until the addition of an extra band does not reduce the information criterion nominated using which.IC. Note that the two-dimensional spatial model is fitted as a random term.

row.corrFitfirst
A logical. If TRUE then, in fitting the model for spatial.model set to corr, the row correlation or variance function is fitted first, followed by the addition of the column correlation or variance function. If FALSE, the order of fitting is reversed.

allow.corrsJointFit
A logical which, if TRUE, will allow the simultaneous fitting of correlation functions for the two dimensions of the grid when separate fits have failed to fit

any correlation functions. This argument is available for when a joint fit hangs the system.

`nugget.variance`

A `logical` which, if `TRUE`, will result in an attempt to fit a nugget or unit-specific variance. Otherwise, a nugget or unit-specific variance will not be fitted.

`dropFixed` A single `character` string or a `character` vector of strings with an element for each level of `sections` in the same order as the `sections` levels. Each string, which if it is not `NA` and after the addition of `". ~ . -"` and conversion to a formula that is then expanded, specifies the sum of a set of terms to be dropped from the fixed formula in fitting splines (`TPPS` and `TPNCSS`). The result is that the fitted model supplied in the `asrtests.obj`, that includes these terms, will be compared with the fitted model that has had them removed and a spatial model added.

An element that is `NA` indicates that no term pertaining to the corresponding `sections` level is to be removed. If `sections` is not `NULL` and a single `character` string has been supplied, the terms specified by the string are taken to be terms that are independent of the `sections` and will be removed when adding the spatial model for the first `sections`.

The terms must match those in the `wald.tab` component of the `asrtests.obj`. The fixed terms will be reordered so that single-variable terms come first, followed by two-variable terms and so on. Note also that multiple terms specified using a single `asreml::at` function can only be dropped as a whole. If the term was specified using an `asreml::at` function with a single level, then it can be removed and either the level itself or its `numeric` position in the levels returned by the `levels` function can be specified.

`dropRandom` A single `character` string or a `character` vector of strings with an element for each level of `sections` in the same order as the `sections` levels. Each string, which if it is not `NA` and after the addition of `" ~ . -"` and conversion to a formula that is then expanded, specifies the sum of a set of terms to be dropped from the random formula in fitting splines (`TPPS` and `TPNCSS`). The result is that the fitted model supplied in the `asrtests.obj`, that includes these terms, will be compared with the fitted model that has had them removed and a spatial model added.

An element that is `NA` indicates that no term pertaining to the corresponding `sections` level is to be removed. If `sections` is not `NULL` and a single `character` string has been supplied, the terms specified by the string are taken to be terms that are independent of the `sections` and will be removed when adding the spatial model for the first `sections`.

The terms must match those in the `vparameters` component of the `asreml.obj` component in the `asrtests.obj`. Note also that multiple terms specified using a single `asreml::at` function can only be dropped as a whole. If the term was specified using an `asreml::at` function with a single level, then it can be removed and either the level itself or its `numeric` position in the levels returned by the `levels` function can be specified.

`nsegs` A pair of `numeric` values giving the number of segments into which the column and row ranges are to be split, respectively, for fitting a P-spline model (`TPPS`) (each value specifies the number of internal knots + 1). If not specified, then

(number of unique values - 1) is used in each dimension; for a grid layout with equal spacing, this gives a knot at each data value. If `sections` is not `NULL` and the grid differs between the `sections`, then `nsegs` will differ between the `sections`.

`nestorder` A `numeric` of length 2. The order of nesting for column and row dimensions, respectively, in fitting a P-spline model (`TPPS`). A value of 1 specifies no nesting, a value of 2 generates a spline with half the number of segments in that dimension, etc. The number of segments in each direction must be a multiple of the order of nesting.

`usRandLinCoeffs`
A `logical` which, if `TRUE` and a P-spline (TPPS) is being fittted, will attempt to fit an unstructured variance model to the constant and linear terms in the interactions for constant and linear terms in one grid dimension interacting with smoooth terms in the second grid dimension. The unstructured variance model can only be fitted if both the constant and linear interaction terms have been retained in the fitted model. This argument can be used to omit the attempt to fit an unstructured variance model when the attempt results in a stystem error.

`rotateX` A `logical` indicating whether to rotate the eigenvectors of the penalty matrix, as described by Piepho, Boer and Williams (2022), when fitting a P-spline (`TPPS`). Setting `rotateX` to `TRUE` results in a a search for an optimized rotation under a model that omits the random spline interaction terms. If `ngridangles` is set to `NULL`, the optimal rotation us found using an optimizer (`nloptr::bobyqa`). Otherwise, the optimal rotation is found by exploring the fit over a two-dimensional grid of rotation angle pairs. The optimization seeks to optimize the criterion nominated in `which.rotacriterion`. Rotation of the eigenvectors is only relevant for `difforder` values greater than 1 and has only been implemented for `difforder` equal to 2.

`ngridangles` A `numeric` of length 2. If `NULL` (the default), the optimal pair of angles for rotating the eignevectors of the penalty matrix of a P-spline (`TPPS`) will be determined using a nonlinear optimizer (`nloptr::bobyqa`). Otherwise, its two values specify the numbers of angles between 0 and 90 degrees for each of the row and column dimensions to be used in determining the optimal pair of angles. Specifying factors of 90 will result in integer-valued angles. The number of grid points, and hence re-analyses will be the product of the values of (`ngridangles` + 1).

`which.rotacriterion`
A single `character` string nominating which of the criteria, out of the `deviance`, the `likelihood`, the `AIC` and the `BIC`, is to be used in determining the optimal rotation of the eigenvectors of the penalty matrix. The `deviance` uses the REML value computed by `asreml`; the other criteria use the full likelihood, evaluated using the REML estimates, that is computed by `infoCriteria.asreml`.

`nrotacores` A `numeric` specifying the number of cores to deploy for running the analyses required to search the two-diemsional grid of rotation angles when `rotateX` is `TRUE`. Parallel processing has been implemented for analyzing, for each column angle, the set of angles to be investigated for the row dimension. The default value of one means that parallel processing will not be used. The value chosen for `nrotacores` needs to balanced against the other processes that are using parallel processing at the same time.

asreml.option
: A single character string specifying whether the grp or mbf methods are to be used to supply externally formed covariate matrices to asreml when fitting a P-spline (TPPS). Compared to the mbf method, the grp method is somewhat faster, but creates large asrtests.objects for which the time it takes to save them can exceed any gains in execution speed. The grp method adds columns to the data.frame containing the data. On the other hand, the mbf method adds only the fixed covariates to data and stores the random covariates in the environment of the internal function that calls the spline-fitting function; there are three smaller data.frames for each section that are not stored in the asreml.object resulting from the fitted model.

tpps4mbf.obj
: An object made with makeTPPSplineMats.data.frame that contains the spline basis information for fitting P-splines. The argument tpps4mbf.obj only needs to be set when the mbf option of asreml.option is being used and it is desired to use mbf data.frames that have been created and stored prior to calling chooseSpatialModelOnIC.asrtests. If tpps4mbf.obj is NULL, makeTPPSplineMats.data.frame will be called internally to produce the required mbf data.frames.

allow.unconverged
: A logical indicating whether to accept a new model even when it does not converge. If FALSE and the fit of the new model does not converge, the supplied asrtests.obj is returned. Also, if FALSE and the fit of the new model has converged, but that of the old model has not, the new model will be accepted.

allow.fixedcorrelation
: A logical indicating whether to accept a new model even when it contains correlations in the model whose values have been designated as fixed, bound or singular. If FALSE and the new model contains correlations whose values have not been able to be estimated, the supplied asrtests.obj is returned. The fit in the asreml.obj component of the supplied asrtests.obj will also be tested and a warning issued if both fixed correlations are found in it and allow.fixedcorrelation is FALSE.

checkboundaryonly
: If TRUE then boundary and singular terms are not removed by rmboundary.asrtests; a warning is issued instead. Note that, for correlation models, the fitting of each dimension and the test for a nugget term are performed with checkboundaryonly set to TRUE and its supplied setting only honoured using a call to rmboundary.asrtests immediately prior to returning the final result of the fitting.

update
: If TRUE, and set.terms is NULL, then newfit.asreml is called to fit the model to be tested, using the values of the variance parameters stored in the asreml.object, that is stored in asrtests.obj, as starting values. If FALSE or set.terms is not NULL, then newfit.asreml will not use the stored variance parameter values as starting values when fitting the new model, the only modifications being (i) to fit aptial terms and (ii) those specified via ....

trace
: If TRUE then the stages in fitting a correlation model are displayed.

which.IC
: A character specifying the information criterion to be used in selecting the best model. Possible values are AIC and BIC. The value of the criterion for supplied model must exceed that for changed model for the changed model to be returned. (For choosing the rotation angle of the eigenvectors of the penalty matrix, see which.rotacriterion.

maxit	A numeric specifying the maximum number of iterations that asreml should perform in fitting a model.
IClikelihood	A character specifying whether Restricted Maximum Likelihood (REML) or the full likelihood, evaluated using REML estimates, (full) are to be used in calculating the information criteria to be included in the test.summary of an asrtests.object or to be used in choosing the best model.
return.asrts	A character string specifying whether the asrtests.object for the "best" fitting model (smallest AIC or BIC), including the supplied nonspatial model, is returned or the asrtests.objects resulting from the attempted fits of "all" of the models specified using trySpatial are returned.
...	Further arguments passed to changeModelOnIC.asrtests, asreml and tpsmmb.

Details

For each spatial model that is to be fitted, a fitted spatial model is only returned if it improves the fit over and above that achieved with the model fit supplied in the asrtests.obj, because terms in the spatial model are not added unless model fit is improved by their addition as measured by an IC. If return.asrts is all, then this applies to each spatial model specified by trySpatial. To force a spatial model to be fitted use addSpatialModel.asrtests. The model fit supplied in the asrtests.obj should not include terms that will be included in any local spatial model. All spatial model terms are fitted as fixed or random. Consequently, the residual model does not have to be iid. The improvement in the fit resulting from the addition of a spatial model to the supplied model is evaluated. Note that the data must be in the order that corresponds to the residual argument with a variable to the right of another variable changing levels in the data frame faster than those of the preceding variables e.g. Row:Column implies that all levels for Column are in consecutive rows of the data.frame that have a single Row level.

For the corr spatial model, the default model is an autocorrelation model of order one (ar1) for each dimension. However, any of the single dimension correlation/variance models from asreml can be specified for each dimension, as can no correlation model for a dimension; the models for the two dimensions can differ. Using a forward selection procedure, a series of models are tried, without removing boundary or singular terms, beginning with the addition of row correlation and followed by the addition of column correlation or, if the row.corrFitfirst is set to FALSE, the reverse order. If the fitting of the first-fitted correlation did not result in a model change because the fitting did not converge or correlations were fixed, but the fit of the second correlation was successful, then adding the first correlation will be retried. If one of the metric correlation functions is specified (e.g. exp), then the row.covar or col.covar will be used in the spatial model. However, because the correlations are fitted separately for the two dimensions, the row.factor and col.factor are needed for all models and are used for any dimension that does not involve a correlation/variance function for the fit being performed. Also, the correlation models are fitted as random terms and so the correlation model will include a variance parameter for the grid even when ar1 is used to specify the correlation model, i.e. the model fitted is a variance model and there is no difference between ar1 and ar1v in fitting the model. The variance parameter for this term represents the spatial variance and the fit necessarily includes a nugget term, this being the residual variance. If any correlation is retained in the model, for a section if sections is not NULL, then the need for a nuggest term is assessed by fixing the corresponding residual variance to one, unless there are multiple residual variances and these are not related to the sections. Once the fitting of the correlation model has been completed, the rmboundary function will be executed with the checkboundaryonly value supplied in the chooseSpatialModelOnIC.asrtests call. Finally, checking for bound and

singular random terms associated with the correlation model and residual terms will be carried out when there are correlation terms in the model and checkboundaryonly has been set to FALSE; as many as possible will be removed from the fitted model, in some cases by fixing variance terms to one.

The tensor-product natural-cubic-smoothing-spline (TPNCSS) spatial model is as described by Verbyla et al. (2018), the tensor-product penalized-cubic-spline (TPPSC2) model with second-order differencing of the penalty is similar to that described by Rodriguez-Alvarez et al. (2018), and the tensor-product, first-difference-penalty, linear spline (TPPSL1) model is amongst those described by Piepho, Boer and Williams (2022). The fixed terms for the spline models are row.covar + col.covar + row.covar:col.covar and the random terms are spl(row.covar) + spl(col.covar) + dev(row.covar) + dev(col.covar) + spl(row.covar):col.covar + row.covar:spl(col.covar) + spl(row.covar):spl(col.covar), except that spl(row.covar) + spl(col.covar) is replaced with spl(row.covar):int(col.covar) + int(row.covar):spl(col.covar) in the TPPSC2 model, where int(.) indicates an intercept or constant value specific to its argument. For TPPSL1 models, the terms spl(row.covar):col.covar + row.covar:spl(col.covar) are omitted, The supplied model should not include any of these terms. However, any fixed or random main-effect Row or Column term that has been included as an initial model for comparison with a spatial model can be removed prior to fitting the spatial model using dropFixed or dropRandom. For the P-spline models with second-order differencing, the model matrices used to fit the pairs of random terms (i) spl(row.covar):int(col.covar) and spl(row.covar):col.covar and (ii) int(row.covar):spl(col.covar) and row.covar:spl(col.covar) are transformed using the spectral decomposition of their penalty matrices. An unstructured variance model is tried for each of these pairs and retained if it improves the fit. For TPPSC2, it is also possible to optimize the rotation of the null-space eigenvectors of the penalty matrix for each of these random-term pairs (for more information see Piepho, Boer and Williams, 2022). The optimization is achieved either using an optimizer or takes the form of a search over a grid of rotation angles for a reduced model; the fit of the full model with rotation using the optimal rotation angles will only be returned if it improves on the fit of the full, unrotated model.

The TPPSC2 and TPPSL1 models are fitted using functions from the R package TPSbits authored by Sue Welham (2022). There are two methods for supplying the spline basis information produced by tpsmmb to asreml. The grp method adds it to the data.frame supplied in the data argument of the asreml call. The mbf method creates smaller data.frames with the spline basis information in the same environment as the internal function that calls the spline-fitting function. If it is desired to use in a later session, an asreml function, or asrtests function that calls asreml, (e.g. predict.asreml, predictPlus.asreml, or changeTerms.asrtests) on an asreml.object created using mbf terms, then the mbf data.frames will need to be recreated using makeTPPSplineMats.data.frame in the new session, supplying, if there has been rotation of the penalty matrix eigenvectors, the theta values that are returned as the attribute theta.opt of the asreml.obj.

All models utlize the function changeModelOnIC.asrtests to assess the model fit, the information criteria used in assessing the fit being calculated using infoCriteria. Arguments from tpsmmb and changeModelOnIC.asrtests can be supplied in calls to chooseSpatialModelOnIC.asrtests and will be passed on to the relevant function though the ellipses argument (...).

The data for experiment can be divided into sections and an attempt to fit the same spatial model to each is made. The fit may differ for each of the sections, but the fit over all of the sections is assessed. For more detail see sections above.

Each combination of a row.coords and a col.coords does not have to specify a single observation;

for example, to fit a local spatial model to the main units of a split-unit design, each combination would correspond to a main unit and all subunits of the main unit would have the same combination.

Value

A list containing four components: (i) asrts, (ii) spatial.IC, (iii) best.spatial.mod, and (iv) best.spatial.IC.

The component asrts itself holds a list of one or more asrtests.objects, either the best overall out of the supplied model and the spatial models, or, for each spatial model, the best out of the supplied model and that spatial model. Each asrtests.object contains the components: (i) asreml.obj, (ii) wald.tab, and (iii) test.summary. If the asrtests.object is the result of fitting a TPPSC2 model with an exploration of the rotation of the eigenvectors of the penalty matrix for the linear components, then the asreml.obj will have an attribute theta.opt that contains the optimal rotation angles of the eigenvectors.

The spatial.IC component holds a data.frame with summary of the values of the information criteria for the supplied model and those resulting from adding the spatial models to the supplied model. In the case of a spatial correlation model, the information criteria for the selected spatial correlation model is returned. If a spatial model could not be fitted, then all returned values will be NA).

The best.spatial.mod component is a character giving the name of the best spatial model, and best.spatial.AIC gives the value of its AIC.

Author(s)

Chris Brien

References

Piepho, H.-P., Boer, M. P., & Williams, E. R. (2022). Two-dimensional P-spline smoothing for spatial analysis of plant breeding trials. *Biometrical Journal*, **64**, 835-857.

Rodriguez-Alvarez, M. X., Boer, M. P., van Eeuwijk, F. A., & Eilers, P. H. C. (2018). Correcting for spatial heterogeneity in plant breeding experiments with P-splines. *Spatial Statistics*, **23**, 52-71.

Verbyla, A. P., De Faveri, J., Wilkie, J. D., & Lewis, T. (2018). Tensor Cubic Smoothing Splines in Designed Experiments Requiring Residual Modelling. *Journal of Agricultural, Biological and Environmental Statistics*, **23(4)**, 478-508.

Welham, S. J. (2022) TPSbits: *Creates Structures to Enable Fitting and Examination of 2D Tensor-Product Splines using ASReml-R.* Version 1.0.0.

See Also

as.asrtests, makeTPPSplineMats.data.frame, addSpatialModelOnIC.asrtests, addSpatialModel.asrtests, changeModelOnIC.asrtests, changeTerms.asrtests, rmboundary.asrtests, testranfix.asrtests, testresidual.asrtests, newfit.asreml, reparamSigDevn.asrtests, changeTerms.asrtests, infoCriteria.asreml

Examples

```
## Not run:

data(Wheat.dat)

#Add row and column covariates
Wheat.dat <- within(Wheat.dat,
                    {
                      cColumn <- dae::as.numfac(Column)
                      cColumn <- cColumn  - mean(unique(cColumn))
                      cRow <- dae::as.numfac(Row)
                      cRow <- cRow - mean(unique(cRow))
                    })

#Fit initial model
current.asr <- asreml(yield ~ Rep + WithinColPairs + Variety,
                      random = ~ Row + Column,
                      data=Wheat.dat)

#Create an asrtests object, removing boundary terms
current.asrt <- as.asrtests(current.asr, NULL, NULL,
                            label = "Random Row and Column effects")
current.asrt <- rmboundary(current.asrt)

# Choose the best of four models for the local spatial variation
current.asrt <- chooseSpatialModelOnIC(current.asrt,
                                       row.covar = "cRow", col.covar = "cColumn",
                                       dropRowterm = "Row", dropColterm = "Column",
                                       asreml.option = "grp")

## End(Not run)
```

`convAsremlobj.asreml` *Recreates an* `asreml` *object so that it is compatible with the currently loaded* `asreml` *version.*

Description

Recreate an existing `asreml` object so that it is compatible with the currently loaded `asreml` version. It involves refitting the model stored in the `asreml` object.

Usage

```
## S3 method for class 'asreml'
convAsremlobj(asreml.obj, ...)
```

Arguments

asreml.obj	An asreml object with a component named call (from a previous call to either asreml or update.asreml).
...	Provision for passing arguments to functions called internally - not used at present.

Value

An asreml object.

Author(s)

Chris Brien

References

Butler, D. G., Cullis, B. R., Gilmour, A. R., Gogel, B. J. and Thompson, R. (2023). *ASReml-R Reference Manual Version 4.2*. VSN International Ltd, https://asreml.kb.vsni.co.uk/.

See Also

newfit.asreml, update.asreml

Examples

```
## Not run:
    m1.asr <- convAsremlobj(m1.asr)

## End(Not run)
```

---

convEffectNames2DataFrame.asreml

Converts the effects names for a term stored in the component of an asreml *object into a* data.frame.

Description

Converts the effects names for a term stored in the component of an asreml object into a data.frame that has a column for each factor and variable in the term. It facilitates adding the effects to the data.frame supplied to asreml for an analysis. This function can only be used with asreml v4.2 or later.

Usage

```
## S3 method for class 'asreml'
convEffectNames2DataFrame(asreml.obj, term, use = "design.matrix", sep = ":", ...)
```

Arguments

asreml.obj	An object resulting from the fitting of a model using asreml v4.2.
term	A character nominating a single model term. It should be the name of the term in the component, of the asreml.obj, nominated in the use argument. The variables/factors in term must match those in this component. Thus, it will not include asreml variance or correlation functions, but may include functions such at at. Generally, each element of the term, separated from other elements by colons, should involve a single factor or variable.
use	A character specifying a component in the asreml.obj. The default is the design.matrix. Other possibilities are the random.coeffs or G.aom. For the option design.matrix, the design argument of asreml.options must be set to TRUE, before executing either the asreml or update.asreml call produces the asreml.obj. The option G.aom specifies the G component of the aom component that is included in the asreml.obj if the aom argument is set to TRUE in either asreml.options before calling asreml or update.asreml, or in a call to asreml or update.asreml that is involved in producing the asreml.obj.
sep	A character specifying the separator of factors/variables in the term.
...	Provision for passing arguments to functions called internally - not used at present.

Value

A data.frame with columns for the factors and variables in term. It includes the attribute effect.names that contains the extracted effects names for the term

Author(s)

Chris Brien

References

Butler, D. G., Cullis, B. R., Gilmour, A. R., Gogel, B. J. and Thompson, R. (2023). *ASReml-R Reference Manual Version 4.2.* VSN International Ltd, https://asreml.kb.vsni.co.uk/.

Examples

```
## Not run:
    G.dat <- convEffectNames2DataFrame(m1.asr, term = "Row:Column", use = "G.aom")

    G.dat <- lapply(c("at(Smarthouse, 'SW'):Lane:Position",
                      "at(Smarthouse, 'SE'):Lane:Position"),
                    function(term, asreml.obj)
                      tmp <- convEffectNames2DataFrame.asreml(asreml.obj, term = term),
                    asreml.obj = m1.asr)
    G.dat <- do.call(rbind, G.dat)

## End(Not run)
```

`estimateV.asreml` *Forms the estimated variance, random or residual matrix for the observations from the variance parameter estimates.*

Description

Forms the estimated variance (**V**), random (**G**) or (**R**) matrix for the observations, a square symmetric matrix of order equal to the number of observations. The estimates of the variance parameters and the information about the `random` and `residual` models for which they were estimated are obtained from the `asreml` object. This function is not available in ASReml-R version 3.

Usage

```
## S3 method for class 'asreml'
estimateV(asreml.obj, which.matrix = "V",
          extra.matrix = NULL, ignore.terms = NULL, fixed.spline.terms = NULL,
          bound.exclusions = c("F","B","S","C"), ...)
```

Arguments

`asreml.obj`	An `asreml` object from a call to `asreml` in which the `data` argument has been set.
`which.matrix`	A `character` giving the matrix that is to be formed. It must be one of `"V"`, to produce the variance matrix $\mathbf{V} = \mathbf{G} + \mathbf{R}$, `"G"` to produce the matrix **G**, corresponding to the `random` formula, or `"R"` to produce the matrix **R**, corresponding to the `residual` formula.
`extra.matrix`	A `matrix` of order equal to the number of observations that is to be added to the `matrix` specified by `which.matrix`, the latter based on the information in `asreml.obj`. It is assumed that the sigma-parameterized values of the variance parameter estimates, such as is given in the `varcomp` component of `summary.asreml`, have been used in calculating `extra.matrix`; the values in the `vparameters` component of `G.param` and `R.param` may be either gamma- or sigma-parameterized. The argument `extra.matrix` can be used in conjunction with `ignore.terms` as a workaround to include components of the variance matrix for variance functions that have not been implemented in `estimateV`.
`ignore.terms`	A `character` giving terms from either the `random` or `residual` models that are to be ignored in that their contributions to the variance is not to be included in the estimated matrix. The term names are those given in the `vparameters` component of the `asreml` object or the `varcomp` component produced by `summary.asreml`, but only up to the first exclamation mark (`!`). This can be used in conjunction with `estimateV.asreml` as a workaround to include components of the variance matrix for variance functions that have not been implemented in `estimateV`.
`fixed.spline.terms`	A `character` vector giving one or more spline terms in the `random` model that are regarded as fixed and so are to be ignored because they are not regarded as contributing to the variance. The term names are those given in the

vparameters component of the asreml object or the varcomp component produced by summary.asreml, but only up to the first exclamation mark (!).

bound.exclusions
: A character specifying one or more bound codes that will result in a variance parameter in the random model being excluded from contributing to the variance. If set to NULL then none will be excluded.

...
: Provision for passing arguments to functions called internally - not used at present.

Details

The information about the variance parameters in the fitted mixed model are obtained from the G.param and R.param components of the asreml object. The function can deal with the following variance functions in either the random or residual models: id, diag, us, ar1, ar2, ar3, sar,sar2, ma1, ma2, arma, exp, gau, cor, corb and corg. All of these functions, except us, can be combined with either v or h. It will also cope with the following functions in the random model: at, str, spl, dev, grp, fa and rr. Additionally, it can deal with the function dsum in the residual model. For further information see the ASReml-R User Guide Version 4 (Butler et al., 2023).

Value

A matrix containing the estimated variance matrix. It has an attribute missing.termmatrix (use attr(x, which = "missing.termmatrix") to access the atrribute). It will be NULL, unless the design matrix could not be obtained for one or more model terms. If is is not NULL, it will be a list of terms that could not be produced for inclusion in the variance matrix estimate, and NA will be returned for the estimated variance matrix.

Author(s)

Chris Brien

References

Butler, D. G., Cullis, B. R., Gilmour, A. R., Gogel, B. J. and Thompson, R. (2023). *ASReml-R Reference Manual Version 4.2.* VSN International Ltd, https://asreml.kb.vsni.co.uk/.

See Also

asreml, simulate.asreml, variofaces.asreml.

Examples

```
## Not run:
data(Wheat.dat)
current.asr <- asreml(yield ~ Rep + WithinColPairs + Variety,
                      random = ~ Row + Column + units,
                      residual = ~ ar1(Row):ar1(Column),
                      data=Wheat.dat)
# Form variance matrix based on estimated variance parameters
V <- estimateV(current.asr)
```

```
## End(Not run)
```

`exploreLSDs.alldiffs` *Explores the computed LSD values for pairwise differences between predictions.*

Description

Given an `alldiffs.object` with an `sed` component, the LSDs are calculated for all pairwise comparisons of predictions. It then calculates (i) a table of frequencies of the LSD values, (ii) the distinct values of the LSDs after rounding, (iii) various statistics from the LSD values, (iv) a measure of the accuracy of each of the LSD statistics, (v) the numbers of false positives and false negatives for each of the LSD statistics if pairwise comparisons are based on the LSD statistic, (vi) the accuracy of each statistic in representing the LSD values for each prediction and (vii) a matrix containing the LSD values for comparing each pair of predictions. Histograms of the frequencies can also be produced.

Usage

```
## S3 method for class 'alldiffs'
exploreLSDs(alldiffs.obj, LSDtype = "overall", LSDby = NULL,
            LSDaccuracy = "maxAbsDeviation", alpha = 0.05, digits = 3,
            retain.zeroLSDs = FALSE,
            zero.tolerance = .Machine$double.eps ^ 0.5,
            plotHistogram = FALSE, ...)
```

Arguments

`alldiffs.obj`	An `alldiffs.object`.
`LSDtype`	A `character` string that can be `overall` or `factor.combinations`. It determines whether the LSD values that are investigated and stored are (i) the `overall` minimum, quantile10, quantile25, mean, median, quantile75, quantile90, and maximum of all pairwise LSDs, or (ii) the minimum, quantile10, quantile25, mean, median, quantile75, quantile90, and maximum for the pairwise LSDs for each `factor.combination`, unless there is only one prediction for a `factor.combination`, when notional LSDs are calculated. The `LSDtype` specified here does not have to match that used in the creating the `alldiffs.object`. See `LSD.frame` for further information on how the LSD statistics are calculated.
`LSDby`	A `character` (vector) of variables names, being the names of the `factor`s or `numeric`s in the `classify`; for each combination of the values the of the `factor`s and `numeric`s, the LSD statistics and accuracy are computed, as well histograms plotted, when `LSDtype` is `factor.combinatons`. The `LSDby` specified here does not have to match that used in the creating the `alldiffs.object`.

`LSDaccuracy` A `character` nominating one of `maxAbsDeviation`, `maxDeviation`, `q90Deviation` or `RootMeanSqDeviation` as the statistic to be calculated as a measure of the accuracy of an LSD statistic when its values are used as an approximate LSD. The option `q90Deviation` produces the sample quantile corresponding to a probability of 0.90. The deviations are the differences between a set of LSDs and an LSD statistic calculated from those LSDs; the accuracy is expressed as a proportion of the value of the LSD statistic.

`alpha` A `numeric` specifying the significance level for an LSD to compare a pair of predictions.

`digits` A `numeric` specifying the number of significant digits to retain in rounding the LSDs before determining the distinct rounded LSDs.

`retain.zeroLSDs` A `logical` indicating whether to retain or omit LSDs that are zero when calculating the summaries of LSDs.

`zero.tolerance` A `numeric` specifying the value such that if an LSD is less than it, the LSD will be considered to be zero.

`plotHistogram` A `logical` indicating whether or not histograms of the LSD values are to be plotted. The `LSDtype` argument controls whether one histogram of all LSD values is plotted or histograms are plotted for each combination of the levels of the factors specified by the `LSDby` argument.

`...` Provision for passing arguments to functions called internally - not used at present.

Details

The false positives and negatives are computed by comparing, for each pair of predictions within each levels-combination of the `LSDby` variables, the significance of the pair difference determined using (i) the true LSD that is computed from the standard error of differences for the pair and (ii) the approximate LSD that is a statistic computed from the true LSDs for all pairwise difference within each levels-combination of the `LSDby` variables. The number of false positives is the number of pairwise differences for which a difference is declared significant using the approximate LSD, but not using the true LSD. The number of false negatives is the number of pairwise differences for which a difference is declared nonsignificant using the approximate LSD, but significant using the true LSD.

The LSD accuracy for a set of LSDs is a function of the deviations of those LSDs and an LSD statistic calculated from them; the accuracy is expressed as a proportion of the value of the LSD statistic.

Value

A `list` with components `frequencies`, `distinct.vals`, `statistics`, `accuracy`, `per.pred.accuracy` and `LSD`:

1. `frequencies` is a `data.frame` with the frequency distribution of the LSD values;
2. `distinct.vals` is a `list`, each component of which contains the distinct values of the LSDs after rounding;

3. statistics is a data.frame with the minimum, quantile10, quantile25, mean, median, quantile75, quantile90, and maximum of the LSD values;
4. accuracy is a data.frame with the accuracies of the minimum, quantile10, quantile25, mean, median, quantile75, quantile90, and maximum of the LSD values with respect to the values from which these statistics are calculated;
5. false.pos is a data.frame with the numbers of false positives for the pairwise comparisons within each levels-combination of the LSDby variables when each of the minimum, quantile10, quantile25, mean, median, quantile75, quantile90, and maximum of the LSD values is used as an approximate LSD in determining the significance of the pairwise differences;
6. false.neg is a data.frame with the numbers of false negatives for the pairwise comparisons within each levels-combination of the LSDby variables when each of the minimum, quantile10, quantile25, mean, median, quantile75, quantile90, and maximum of the LSD values is used as an approximate LSD in determining the significance of the pairwise differences;
7. per.pred.accuracy is a data.frame with the accuracies of the minimum, quantile10, quantile25, mean, median, quantile75, quantile90, and maximum of the LSD values for a set of predictions when these statistics are used to represent the LSDs for the comparisons amongst the set of predictions;
8. LSD is a square matrix containing the LSD values for all pairwise comparisons of the predictions contained in the supplied alldiffs.obj.

In the statistics, accuracy, false.pos and false.neg data.frames, c is the number of pairwise comparisons on which the values in the same row are based. The accuracy measure is specified by the LSDaccuracy argument.

Author(s)

Chris Brien

See Also

asremlPlus-package, plotLSDs.data.frame, plotLSDs.alldiffs, plotLSDerrors.alldiffs, plotLSDerrors.data.frame, recalcLSD.alldiffs, pickLSDstatistics.alldiffs, redoErrorIntervals.alldiffs

Examples

```
data(WaterRunoff.dat)

##Use asreml to get predictions and associated statistics

## Not run:
asreml.options(keep.order = TRUE) #required for asreml-R4 only
current.asr <- asreml(fixed = pH ~ Benches + (Sources * (Type + Species)),
                      random = ~ Benches:MainPlots,
                      keep.order=TRUE, data= WaterRunoff.dat)
current.asrt <- as.asrtests(current.asr, NULL, NULL)
TS.diffs <- predictPlus(classify = "Sources:Type",
                        asreml.obj = current.asr,
                        wald.tab = current.asrt$wald.tab,
```

```
                        present = c("Sources", "Type", "Species"))

## End(Not run)

## Use lmeTest and emmmeans to get predictions and associated statistics

if (requireNamespace("lmerTest", quietly = TRUE) &
    requireNamespace("emmeans", quietly = TRUE))
{
  m1.lmer <- lmerTest::lmer(pH ~ Benches + (Sources * (Type + Species)) +
                              (1|Benches:MainPlots),
                            data=na.omit(WaterRunoff.dat))
  TS.emm <- emmeans::emmeans(m1.lmer, specs = ~ Sources:Type)
  TS.preds <- summary(TS.emm)
  den.df <- min(TS.preds$df, na.rm = TRUE)
  ## Modify TS.preds to be compatible with a predictions.frame
  TS.preds <- as.predictions.frame(TS.preds, predictions = "emmean",
                                   se = "SE", interval.type = "CI",
                                   interval.names = c("lower.CL", "upper.CL"))

  ## Form an all.diffs object and check its validity
  TS.vcov <- vcov(TS.emm)
  TS.diffs <- allDifferences(predictions = TS.preds, classify = "Sources:Type",
                             vcov = TS.vcov, tdf = den.df)
  validAlldiffs(TS.diffs)
}

## Plot p-values for predictions obtained using asreml or lmerTest
if (exists("TS.diffs"))
{
  ##Explore the LSD values for predictions obtained using asreml or lmerTest
  LSDstat <- exploreLSDs(TS.diffs, LSDtype = "factor.combinations",
                         LSDby = "Sources")
}
```

`facCombine.alldiffs` *Combines several factors into one in the components of an* `alldiffs.object`

Description

Combines several `factor`s, in the `prediction` component of `object`, into one whose `levels` are the combinations of the used `levels` of the individual `factor`s. The matching changes are made to the other components and the attributes of the `alldiffs.object`. If any of the factors to be combined are in `LSDby`, they are removed from the `LSDby`, unless the factors to be combined are exactly those in the `LSDby`. The levels of the `factors` are combined using `fac.combine` from the `dae` package.

Usage

```
## S3 method for class 'alldiffs'
facCombine(object, factors, order="standard",
           combine.levels=TRUE, sep="_", level.length = NA, ...)
```

Arguments

object	An alldiffs.object.
factors	A character containing the names of factors in the prediction component of object whose levels are to be combined.
order	Either standard or yates. The order in which the levels combinations of the factors are to be considered as numbered when forming the levels of the combined factor; standard numbers them as if they are arranged in standard order, that is with the levels of the first factor moving slowest and those of the last factor moving fastest; yates numbers them as if they are arranged in Yates order, that is with the levels of the first factor moving fastest and those of the last factor moving slowest.
combine.levels	A logical specifying whether the levels labels of the new factor are to be combined from those of the factors being combined. The default is to use the integers from 1 to the product of the numbers of combinations of used levels of the individual factors, numbering the levels according to order.
sep	A character string to separate the levels when combine.levels = TRUE.
level.length	The maximum number of characters from the levels of factors to use in the row and column labels of the tables of pairwise differences and their p-values and standard errors.
...	Further arguments passed to redoErrorIntervals.alldiffs.

Value

A modified alldiffs.object.

Author(s)

Chris Brien

See Also

as.alldiffs, allDifferences.data.frame, print.alldiffs, sort.alldiffs, renewClassify.alldiffs; fac.combine in package **dae**.

Examples

```
data("Ladybird.dat")

## Use asreml to get predictions and associated statistics

## Not run:
m1.asr <- asreml(logitP ~ Host*Cadavers*Ladybird,
```

```
                  random = ~ Run,
                  data = Ladybird.dat)
  current.asrt <- as.asrtests(m1.asr)
  HCL.pred <- asreml::predict.asreml(m1.asr, classify="Host:Cadavers:Ladybird",
                                     sed=TRUE)
  HCL.preds <- HCL.pred$pvals
  HCL.sed <- HCL.pred$sed
  HCL.vcov <- NULL
  wald.tab <-  current.asrt$wald.tab
  den.df <- wald.tab[match("Host:Cadavers:Ladybird", rownames(wald.tab)), "denDF"]

## End(Not run)

  ## Use lmeTest and emmmeans to get predictions and associated statistics
  if (requireNamespace("lmerTest", quietly = TRUE) &
      requireNamespace("emmeans", quietly = TRUE))
  {
    m1.lmer <- lmerTest::lmer(logitP ~ Host*Cadavers*Ladybird + (1|Run),
                              data=Ladybird.dat)
    HCL.emm <- emmeans::emmeans(m1.lmer, specs = ~ Host:Cadavers:Ladybird)
    HCL.preds <- summary(HCL.emm)
    den.df <- min(HCL.preds$df)
    ## Modify HCL.preds to be compatible with a predictions.frame
    HCL.preds <- as.predictions.frame(HCL.preds, predictions = "emmean",
                                      se = "SE", interval.type = "CI",
                                      interval.names = c("lower.CL", "upper.CL"))
    HCL.vcov <- vcov(HCL.emm)
    HCL.sed <- NULL
  }

  ## Use the predictions obtained with either asreml or lmerTest
  if (exists("HCL.preds"))
  {
    ## Form an all.diffs object
   HCL.diffs <- as.alldiffs(predictions = HCL.preds, classify = "Host:Cadavers:Ladybird",
                            sed = HCL.sed, vcov = HCL.vcov, tdf = den.df)

    ## Check the class and validity of the alldiffs object
    is.alldiffs(HCL.diffs)
    validAlldiffs(HCL.diffs)

    ## Combine Cadavers and Ladybird
    HCL.diffs <- facCombine(HCL.diffs, factors = c("Cadavers","Ladybird"))

    ## Check the validity of HCL.diffs
    validAlldiffs(HCL.diffs)
  }
```

facRecast.alldiffs *Reorders and/or revises the factor levels using the order of old levels in* levels.order *and the new labels for the levels given in* newlabels. *The values in* levels.order *must be unique.*

Description

Reorders and revises the levels and labels of a factor, in the prediction component of an alldiffs.object. The values in the levels.order vector should be the same as the levels in the existing factor, but the order can be changed. To revise the levels, specify the new levels in the newlabels vector and these will replace the corresponding value in the levels.order vector. The matching changes are made to the other components and attributes of the alldiffs.object.

Usage

```
## S3 method for class 'alldiffs'
facRecast(object, factor, levels.order = NULL, newlabels = NULL, ...)
```

Arguments

object	An alldiffs.object.
factor	A character containing the name of a factor in the prediction component of object whose levels and labels are to be recast.
levels.order	A vector of length levels(factor) containing the old levels in the new order for the factor being created. If levels.order is NULL, then the current levels of levels(factor) are used.
newlabels	A vector of length levels(factor) containing values to use in the revision.
...	Further arguments passed to the factor call creating the new factor.

Value

A modified alldiffs.object.

Author(s)

Chris Brien

See Also

as.alldiffs, allDifferences.data.frame, print.alldiffs, sort.alldiffs, facCombine.alldiffs, facRename.alldiffs, renewClassify.alldiffs; fac.recast in package **dae**.

Examples

```
  data("Ladybird.dat")

  ## Use asreml to get predictions and associated statistics

  ## Not run:
  m1.asr <- asreml(logitP ~ Host*Cadavers*Ladybird,
                   random = ~ Run,
                   data = Ladybird.dat)
  current.asrt <- as.asrtests(m1.asr)
  HCL.pred <- asreml::predict.asreml(m1.asr, classify="Host:Cadavers:Ladybird",
                                     sed=TRUE)
  HCL.preds <- HCL.pred$pvals
  HCL.sed <- HCL.pred$sed
  HCL.vcov <- NULL
  wald.tab <-  current.asrt$wald.tab
  den.df <- wald.tab[match("Host:Cadavers:Ladybird", rownames(wald.tab)), "denDF"]

## End(Not run)

  ## Use lmeTest and emmmeans to get predictions and associated statistics
  if (requireNamespace("lmerTest", quietly = TRUE) &
      requireNamespace("emmeans", quietly = TRUE))
  {
    m1.lmer <- lmerTest::lmer(logitP ~ Host*Cadavers*Ladybird + (1|Run),
                              data=Ladybird.dat)
    HCL.emm <- emmeans::emmeans(m1.lmer, specs = ~ Host:Cadavers:Ladybird)
    HCL.preds <- summary(HCL.emm)
    den.df <- min(HCL.preds$df)
    ## Modify HCL.preds to be compatible with a predictions.frame
    HCL.preds <- as.predictions.frame(HCL.preds, predictions = "emmean",
                                      se = "SE", interval.type = "CI",
                                      interval.names = c("lower.CL", "upper.CL"))
    HCL.vcov <- vcov(HCL.emm)
    HCL.sed <- NULL
  }

  ## Use the predictions obtained with either asreml or lmerTest
  if (exists("HCL.preds"))
  {
    ## Form an all.diffs object
   HCL.diffs <- allDifferences(predictions = HCL.preds, classify = "Host:Cadavers:Ladybird",
                               sed = HCL.sed, vcov = HCL.vcov, tdf = den.df)

    ## Check the class and validity of the alldiffs object
    is.alldiffs(HCL.diffs)
    validAlldiffs(HCL.diffs)

    ## Recast the Ladybird and Host factors
    HCL.diffs <- facRecast(HCL.diffs, factor = "Ladybird",
                           newlabels = c("none", "present"))
    HCL.diffs <- facRecast(HCL.diffs, factor = "Ladybird",
```

```
                          levels.order = c("present", "none"),
                          newlabels = c("yes","no"))
  HCL.diffs <- facRecast.alldiffs(HCL.diffs, factor = "Host",
                                  levels.order = c("trefoil", "bean"))

  ## Check the validity of HCL.diffs
  validAlldiffs(HCL.diffs)
}
```

`facRename.alldiffs` *Renames* `factor`*s in the* `prediction` *component of an* `alldiffs.object`.

Description

Renames `factor`s in the `prediction` component of an `alldiffs.object`. These changes are propagated to the other components and attributes of the `alldiffs.object`.

Usage

```
## S3 method for class 'alldiffs'
facRename(object, factor.names, newnames,  ...)
```

Arguments

`object`	An `alldiffs.object`.
`factor.names`	A `character` containing the names of the `factor`s in the `prediction` component of `object` that are to be renamed.
`newnames`	A `character` containing the new names of the `factor`s in the `prediction` component of `object`.
`...`	Provision for passing arguments to functions called internally - not used at present.

Value

A modified `alldiffs.object`.

Author(s)

Chris Brien

See Also

`as.alldiffs`, `allDifferences.data.frame`, `print.alldiffs`, `sort.alldiffs`, `facCombine.alldiffs`, `facRecast.alldiffs`, `renewClassify.alldiffs`; `fac.recast` in package **dae**.

Examples

```
data("Ladybird.dat")

## Use asreml to get predictions and associated statistics

## Not run:
m1.asr <- asreml(logitP ~ Host*Cadavers*Ladybird,
                 random = ~ Run,
                 data = Ladybird.dat)
current.asrt <- as.asrtests(m1.asr)
HCL.pred <- asreml::predict.asreml(m1.asr, classify="Host:Cadavers:Ladybird",
                                   sed=TRUE)
HCL.preds <- HCL.pred$pvals
HCL.sed <- HCL.pred$sed
HCL.vcov <- NULL
wald.tab <-  current.asrt$wald.tab
den.df <- wald.tab[match("Host:Cadavers:Ladybird", rownames(wald.tab)), "denDF"]

## End(Not run)

## Use lmeTest and emmmeans to get predictions and associated statistics
if (requireNamespace("lmerTest", quietly = TRUE) &
    requireNamespace("emmeans", quietly = TRUE))
{
  m1.lmer <- lmerTest::lmer(logitP ~ Host*Cadavers*Ladybird + (1|Run),
                            data=Ladybird.dat)
  HCL.emm <- emmeans::emmeans(m1.lmer, specs = ~ Host:Cadavers:Ladybird)
  HCL.preds <- summary(HCL.emm)
  den.df <- min(HCL.preds$df)
  ## Modify HCL.preds to be compatible with a predictions.frame
  HCL.preds <- as.predictions.frame(HCL.preds, predictions = "emmean",
                                    se = "SE", interval.type = "CI",
                                    interval.names = c("lower.CL", "upper.CL"))
  HCL.vcov <- vcov(HCL.emm)
  HCL.sed <- NULL
}

## Use the predictions obtained with either asreml or lmerTest
if (exists("HCL.preds"))
{
  ## Form an all.diffs object
  HCL.diffs <- allDifferences(predictions = HCL.preds,
                              classify = "Host:Cadavers:Ladybird",
                              sed = HCL.sed, vcov = HCL.vcov, tdf = den.df)

  ## Check the class and validity of the alldiffs object
  is.alldiffs(HCL.diffs)
  validAlldiffs(HCL.diffs)

  ## Rename Cadavers
 HCL.diffs <- facRename(HCL.diffs, factor.names = "Cadavers", newnames = "Cadaver.nos")
```

```
  ## Check the validity of HCL.diffs
  validAlldiffs(HCL.diffs)
}
```

---

findLSDminerrors.alldiffs

Find LSD values that minimize the number of errors in pairwise comparisons of predictions.

Description

Given an alldiffs.object with an sed component, a search is made of a set of equally spaced values between the minimum and maximum values of the LSDs, calculated from the sed component of the alldiffs.object, to identify LSD values that minimize the number of errors made in deciding on the significance of pairs of predicted values stored in the alldiffs.object. If LSDtype is set to overall, a search is made over the range of LSD values for all pairwise comparisons for a single LSD value; if LSDtype is set to factor.combinations, a separate search is made over the LSD values for the set of pairwise comparisons for each factor.combination in order to identify a single value for each set. The number of values used in the search is controlled by the argument nvalues. For each value in the search, the numbers of false positives and false negatives resulting from employing it as the LSD for each set of pairwise comparisons is calculated. A criterion that combines the false positives and negative is calculated using the false.pos.wt, the criterion being the number of false postives times the false.pos.wt plus the number of false negatives. The value chosen for the LSD is the smallest value from amongst those with the minimum value of the criterion and the least number of false positives. A secondary search with 10 equally spaced values is made of the interval below the chosen value and the search value immediately below it to check whether the chosen grid value can be further reduced without changing the value of either its criterion or the number of false positives.

The primary options for changing the numbers of errors associated with the values resulting from the searching is to manipulate the LSDby and/or false.pos.wt arguments.

Usage

```
## S3 method for class 'alldiffs'
findLSDminerrors(alldiffs.obj,
                 LSDtype = "overall", LSDby = NULL,
                 alpha = 0.05,
                 false.pos.wt = 10, nvalues = 100,
                 retain.zeroLSDs = FALSE,
                 zero.tolerance = .Machine$double.eps ^ 0.5,
                 trace = FALSE, ...)
```

Arguments

alldiffs.obj	An alldiffs.object.

LSDtype	A character string that can be overall or factor.combinations. It determines whether the minimum LSD values that are obtained are those for (i) all pairwise comparisons of the predicted values, unless there is only one, possibly repeated, prediction, when a notional LSD is calculated, or (ii) for each combination of the values of the factors and numerics named in LSDby, unless there is only one prediction for a combination, when notional LSDs are calculated. The LSDtype specified here does not have to match that used in the creating the alldiffs.object.
LSDby	A character (vector) of variables names, being the names of some of the factors or numerics in the classify; for each combination of the values of the factors and numerics, the LSD errors are to be computed when LSDtype is factor.combinatons. The LSDby specified here does not have to match that used in the creating the alldiffs.object.
alpha	A numeric specifying the significance level for an LSD to compare a pair of predictions.
false.pos.wt	A numeric that specifies the weight (e.g. 3 so that a false positive is considered to be equivalent to three false negatives) to apply to the number of false positives in calculating the weighted sums of the numbers of false positives and negatives that is used as the criterion to be minimized in selecting the LSD value that results in the minimum number of errors. If, for a particular LSD value, the number of false positives is p, the number of false negatives is n and w the value of false.pos.wt, then then criterion for that LSD value is $(wp) + n$. The default of 10 for false.pos.wt greatly favours false negatives; a value with one false positive and no false negative will only be chosen over a value with no false positive when the latter value has more than 10 false negatives.
nvalues	A numeric specifying the number of equally spaced LSD values, between the minimum and maximum LSD values for a set of paired comparisons, to be used in the search for the LSD value with the minimum number of errors.
retain.zeroLSDs	A logical indicating whether to retain or omit LSDs that are zero when calculating the numbers of errors.
zero.tolerance	A numeric specifying the value such that if an LSD is less than it, the LSD will be considered to be zero.
trace	A logical indicating whether details of the searching are to be output.
...	Provision for passing arguments to functions called internally - not used at present.

Value

A data.frame containing the chosen LSD(s), its(their) numbers of false positives and negatives and the value(s) of the false criterion.

Author(s)

Chris Brien

See Also

asremlPlus-package, exploreLSDs.alldiffs plotLSDs.data.frame, plotLSDs.alldiffs, plotLSDerrors.alldiffs, plotLSDerrors.data.frame, recalcLSD.alldiffs, redoErrorIntervals.alldiffs

Examples

```
data(WaterRunoff.dat)

##Use asreml to get predictions and associated statistics

## Not run:
asreml.options(keep.order = TRUE) #required for asreml-R4 only
current.asr <- asreml(fixed = pH ~ Benches + (Sources * (Type + Species)),
                      random = ~ Benches:MainPlots,
                      keep.order=TRUE, data= WaterRunoff.dat)
current.asrt <- as.asrtests(current.asr, NULL, NULL)
TS.diffs <- predictPlus(classify = "Sources:Type",
                        asreml.obj = current.asr,
                        wald.tab = current.asrt$wald.tab,
                        present = c("Sources", "Type", "Species"))

## End(Not run)

## Use lmeTest and emmmeans to get predictions and associated statistics

if (requireNamespace("lmerTest", quietly = TRUE) &
    requireNamespace("emmeans", quietly = TRUE))
{
  m1.lmer <- lmerTest::lmer(pH ~ Benches + (Sources * (Type + Species)) +
                              (1|Benches:MainPlots),
                            data=na.omit(WaterRunoff.dat))
  TS.emm <- emmeans::emmeans(m1.lmer, specs = ~ Sources:Type)
  TS.preds <- summary(TS.emm)
  den.df <- min(TS.preds$df, na.rm = TRUE)
  ## Modify TS.preds to be compatible with a predictions.frame
  TS.preds <- as.predictions.frame(TS.preds, predictions = "emmean",
                                   se = "SE", interval.type = "CI",
                                   interval.names = c("lower.CL", "upper.CL"))

  ## Form an all.diffs object and check its validity
  TS.vcov <- vcov(TS.emm)
  TS.diffs <- allDifferences(predictions = TS.preds, classify = "Sources:Type",
                             vcov = TS.vcov, tdf = den.df)
  validAlldiffs(TS.diffs)
}

## Choose LSD values with the minimum mumber of error for pairwise comparisons of
##   the predictions obtained using asreml or lmerTest
if (exists("TS.diffs"))
{
  ##Pick the LSD values for predictions obtained using asreml or lmerTest
```

```
 minLSD <- findLSDminerrors(TS.diffs)
TS.diffs <- redoErrorIntervals(TS.diffs, LSDtype = "supplied", LSDsupplied = minLSD["LSD"])
 TS.diffs$LSD
 minLSDs <- findLSDminerrors(TS.diffs, LSDtype = "factor.combinations",
                             LSDby = "Sources")
 TS.diffs <- redoErrorIntervals(TS.diffs, LSDtype = "supplied",
                                LSDby = "Sources", LSDsupplied = minLSDs["LSD"])
 TS.diffs$LSD
}
```

`getASRemlVersionLoaded` *Finds the version of asreml that is loaded and returns the initial characters in version.*

Description

Checks that `asreml` is loaded and, if it is, returns the first `nchar` characters of the version that is loaded.

Usage

```
getASRemlVersionLoaded(nchar = NULL, notloaded.fault = FALSE)
```

Arguments

`nchar`	The number of characters in the asreml version to get.
`notloaded.fault`	A `logical` indicating whether a fault is to occur if `asreml` is not loaded.

Value

A character, being the first `nchar` characters of the version of `asreml` that is loaded.

Author(s)

Chris Brien

See Also

`loadASRemlVersion`.

Examples

```
## Not run:
getASRemlVersionLoaded()
## End(Not run)
```

getFormulae.asreml *Gets the formulae from an asreml object.*

Description

Gets the formulae nominated in the which argument from the call stored in an asreml object.

Usage

```
## S3 method for class 'asreml'
getFormulae(asreml.obj, which = c("fixed", "random", "residual"),
            expanded = FALSE, envir = parent.frame(), ...)
```

Arguments

asreml.obj	An object resulting from the fitting of a model using asreml.
which	A character listing the formula(e) to be extracted from the call stored in asreml.obj. it should be some combination of fixed, random, residual, sparse and all. If all is included then all formula(e) will be returned, those not having been specified in the call being NULL.
expanded	A logical indicating whether terms are to be expanded to the sum of a set of individual terms.
envir	The environment in which the formula(e) are to be evaluated. May also be NULL, a list, a data.frame, a pairlist or an integer as specified to sys.call.
...	Arguments passed on to update.formula and ultimately to terms.formula.

Value

A list containing a component with each of the extracted formula(e), the name of a component being the formula that it contains.

Author(s)

Chris Brien

See Also

printFormulae.asreml

Examples

```
## Not run:
   data(Wheat.dat)
   current.asr <- asreml(yield ~ Rep + WithinColPairs + Variety,
                         random = ~ Row + Column + units,
                         residual = ~ ar1(Row):ar1(Column),
                         data=Wheat.dat)
```

```
    getFormulae(current.asr)

## End(Not run)
```

`getTestEntry.asrtests` *Gets the entry for a test recorded in the test.summary data.frame of an* `asrtests.object`

Description

Matches the `label` in the `term` column of the `test.summary` `data.frame` in the supplied `asrtests.object` and extracts the line for it. It only matches the last occurrence of `label`.

Usage

```
## S3 method for class 'asrtests'
getTestEntry(asrtests.obj, label, error.absent = TRUE, ...)
```

Arguments

`asrtests.obj`	An `asrtests.object` containing the components (i) `asreml.obj`, (ii) `wald.tab`, and (iii) `test.summary`.
`label`	A `character` specifying the `label` of the test for which the entry is required. If `testranfix.asrtests` was used for the test of interest, then the `label` will be the value of the `term` argument supplied to `testranfix.asrtests`. For `changeModelOnIC.asrtests`, the `label` will be the value of the `label` argument. Other arguments will be relevant for other test and change functions.
`error.absent`	A `logical` indicating whether the absence of the supplied `label` is to result in an error. If set to `FALSE`, `NULL` is returned.
`...`	provision for passing arguments to functions called internally - not used at present.

Value

A one-line `data.frame` containing the entry or, `error.absent` is `NULL`, `NULL`.

Author(s)

Chris Brien

See Also

`getTestPvalue.asrtests`, `as.asrtests`,
`testranfix.asrtests`, `testswapran.asrtests`, `testresidual.asrtests`,
`changeModelOnIC.asrtests`, `changeTerms.asrtests`, `chooseModel.asrtests`

Examples

```
## Not run:
data(Wheat.dat)
current.asr <- asreml(yield ~ Rep + WithinColPairs + Variety,
                      random = ~ Row + Column + units,
                      residual = ~ ar1(Row):ar1(Column),
                      data=Wheat.dat)
current.asrt <- as.asrtests(current.asr, NULL, NULL)
current.asrt <- rmboundary(current.asrt)
# Test nugget term
current.asrt <- testranfix(current.asrt, "units", positive=TRUE)
getTestEntry(current.asrt, label = "units")

## End(Not run)
```

---

getTestPvalue.asrtests

Gets the p-value for a test recorded in the test.summary data.frame of an asrtests.object

---

Description

Matches the label in the term column of the test.summary data.frame in the supplied asrtests.object and extracts its p-value. It only matches the last occurrence of label.

Usage

```
## S3 method for class 'asrtests'
getTestPvalue(asrtests.obj, label, ...)
```

Arguments

asrtests.obj	An asrtests.object containing the components (i) asreml.obj, (ii) wald.tab, and (iii) test.summary.
label	A character specifying the label of the test for which the p-value is required. If testranfix.asrtests was used for the test of interest, then the label will be the value of the term argument supplied to testranfix.asrtests. Other arguments will be relevant for other test functions.
...	provision for passing arguments to functions called internally - not used at present.

Value

An numeric containing the p-value. It can be NA, for example when a p-value could not be calculated.

Author(s)

Chris Brien

See Also

getTestEntry.asrtests, as.asrtests,
testranfix.asrtests, testswapran.asrtests, testresidual.asrtests,
changeTerms.asrtests, chooseModel.asrtests

Examples

```
## Not run:
data(Wheat.dat)
current.asr <- asreml(yield ~ Rep + WithinColPairs + Variety,
                      random = ~ Row + Column + units,
                      residual = ~ ar1(Row):ar1(Column),
                      data=Wheat.dat)
current.asrt <- as.asrtests(current.asr, NULL, NULL)
current.asrt <- rmboundary(current.asrt)
# Test nugget term
current.asrt <- testranfix(current.asrt, "units", positive=TRUE)
getTestPvalue(current.asrt, label = "units")

## End(Not run)
```

---

infoCriteria *Computes AIC and BIC for models.*

Description

Computes Akiake and Bayesian (Schwarz) Information Criteria for models. Either the Restricted Maximum likelihood (`REML`) or the full likelihood (`full`) can be used. The full likelihood, evaluated using REML estimates is used when it is desired to compare models that differ in their fixed models.

Usage

```
## S3 method for class 'asreml'
infoCriteria(object, DF = NULL,
            bound.exclusions = c("F","B","S","C"),
            IClikelihood = "REML", fixedDF = NULL, varDF = NULL, ...)
## S3 method for class 'list'
infoCriteria(object, bound.exclusions = c("F","B","S","C"),
            IClikelihood = "REML", fixedDF = NULL, varDF = NULL, ...)
```

Arguments

object	An asreml object resulting from the fitting of a model using REML or a list of asreml objects. If the components of the list are named, then those names will be used as the rownames for the returned data.frame.
DF	A numeric giving the number of estimated variance parameters. If NULL then this is determined from the information in object. This argument has been replaced by varDF, but is retained for compatibility with legacy code. It is not available with the list method.
bound.exclusions	A character specifying the bound (constraint) codes that will result in a variance parameter being excluded from the count of estimated variance parameters. If set to NULL then none will be excluded.
IClikelihood	A character specifying whether Restricted Maximum Likelihood (REML) or the full likelihood, evaluated using REML estimates, (full) are to be used in calculating the information criteria for family set to asr_gaussian. For family set to asr_binomial or asr_poisson and with dispersion set to 1, the deviance is extracted from object and used to calculate the AIC and BIC (as suggested by Damian Collins); the setting of IClikelihood is ignored and the log-likelihood set to NA. The information criteria are not valid for other settings of family and dispersion.
fixedDF	A numeric giving the number of estimated fixed parameters. If NULL then this is determined from the information in object. For object a list only a single value that is used for all components of the list has been implemented.
varDF	A numeric giving the number of estimated variance parameters. If NULL then this is determined from the information in object. It replaces the DF argument. For object a list only a single value that is used for all components of the list has been implemented.
...	Provision for passing arguments to functions called internally - not used at present.

Details

The variance degrees of freedom (varDF) are the number of number of variance parameters that have been estimated, excluding those whose estimates have a code for bound specified in bound.exclusions. If varDF is not NULL, the supplied value is used. Otherwise varDF is determined from the information in object, i.e. if object is an asreml object then from it, or if object is a list then from each asreml object in the list. Similarly, the fixed degrees of freedom (fixedDF) are the number of number of fixed parameters that have been estimated, any coefficients that have the value NA being excluded. If fixedDF is not NULL, the supplied value is used. Otherwise fixedDF is determined from the information in object.

If ASReml-R version 4 is being used then the codes specified in bound.exclusions are not restricted to a subset of the default codes, but a warning is issued if a code other than these is specified. For ASReml-R version 3, only a subset of the default codes are allowed: F (Fixed), B (Boundary), C (Constrained) and S (Singular).

The calculation of the information criteria is an adaptation of the code supplied in File S1 of Verbyla (2019). The log-likelihood is calculated as loglik = log(REML) - log(|C|)/2, where C is the

inverse coefficient matrix; the term involving **C** is omitted for REML. The AIC is calculated as -2*loglik+2*(varDF+fixedDF) and the BIC as -2*loglik+(fixedDF+varDF)*log(n-r+fixedDF), where n is the number of observations and r is the rank of the fixed effects design matrix. For REML, fixedDF = 0.

Value

A data.frame containing the numbers of estimated fixed (fixedDF) and variance (varDF) parameters, the number of bound parameters (NBound), AIC, BIC and the value of the log-likelihood (loglik). All elements of the data.frame will be set to NA for the invalid combinations of family and dispersion as noted in the IClikelihood argument. If object is a list and its components are named, then those names will be used to set the rownames of the data.frame.

Author(s)

Chris Brien

References

Verbyla, A. P. (2019). A note on model selection using information criteria for general linear models estimated using REML. *Australian & New Zealand Journal of Statistics*, **61**, 39–50. doi:10.1111/anzs.12254.

See Also

REMLRT.asreml, changeTerms.asrtests, changeModelOnIC.asrtests

Examples

```
## Not run:
   data(Wheat.dat)
   ## Fit several models to the wheat data and calculate their ICs
   # Fit initial model
   m.max <- asreml(yield ~ Rep + WithinColPairs + Variety,
                   random = ~ Row + Column + units,
                   residual = ~ ar1(Row):ar1(Column),
                   data=Wheat.dat)
   infoCriteria(m.max.asr, IClikelihood = "full")

   #Drop term for within Column pairs
   m1 <- asreml(yield ~ Rep + Variety,
                random = ~ Row + Column + units,
                residual = ~ ar1(Row):ar1(Column),
                data=Wheat.dat)

   #Drop nugget term
   m2 <- asreml(yield ~ Rep + WithinColPairs + Variety,
                random = ~ Row + Column,
                residual = ~ ar1(Row):ar1(Column),
                data=Wheat.dat)
```

```
#Drop Row autocorrelation
m3 <- asreml(yield ~ Rep + WithinColPairs + Variety,
               random = ~ Row + Column + units,
               residual = ~ Row:ar1(Column),
               data=Wheat.dat)

#Drop Col autocorrelation
m4 <- asreml(yield ~ Rep + WithinColPairs + Variety,
             random = ~ Row + Column + units,
             residual = ~ ar1(Row):Column,
             data=Wheat.dat)

mods.asr <- list(m.max, m1, m2, m3, m4)
infoCriteria(mods.asr, IClikelihood = "full")


## End(Not run)
```

`is.alldiffs` *Tests whether an object is of class alldiffs*

Description

A single-line `function` that tests whether an object is of `class` `alldiffs`.

Usage

```
is.alldiffs(object)
```

Arguments

`object`	An `object` to be tested.

Value

A `logical`.

Author(s)

Chris Brien

See Also

`asremlPlus-package`, `alldiffs.object`, `is.alldiffs`, `as.alldiffs`

Examples

```
data(Oats.dat)

## Use lmerTest and emmmeans to get predictions and associated statistics
if (requireNamespace("lmerTest", quietly = TRUE) &
    requireNamespace("emmeans", quietly = TRUE))
{
  m1.lmer <- lmerTest::lmer(Yield ~ Nitrogen*Variety + (1|Blocks/Wplots),
                            data=Oats.dat)
  Var.emm <- emmeans::emmeans(m1.lmer, specs = ~ Nitrogen:Variety)
  Var.preds <- summary(Var.emm)
  den.df <- min(Var.preds$df)
  ## Modify Var.preds to be compatible with a predictions.frame
  Var.preds <- as.predictions.frame(Var.preds, predictions = "emmean",
                                    se = "SE", interval.type = "CI",
                                    interval.names = c("lower.CL", "upper.CL"))
  Var.vcov <- vcov(Var.emm)
  Var.sed <- NULL

## Form an all.diffs object
 Var.diffs <- as.alldiffs(predictions = Var.preds, classify = "Nitrogen:Variety",
                          sed = Var.sed, vcov = Var.vcov, tdf = den.df)

## check the class of Var.diffs
is.alldiffs(Var.diffs)
}
```

`is.asrtests` *Tests whether an object is of class asrtests*

Description

A single-line `function` that tests whether an object is of `class` `asrtests`.

Usage

```
is.asrtests(object)
```

Arguments

`object`	An `object` to be tested.

Value

A `logical`.

Author(s)

Chris Brien

See Also

asremlPlus-package, asrtests.object, is.asrtests, as.asrtests

Examples

```
## Not run:
library(dae)
library(asreml)
library(asremlPlus)
## use ?Wheat.dat for data set details
data(Wheat.dat)

# Fit initial model
current.asr <- asreml(yield ~ Rep + WithinColPairs + Variety,
                      random = ~ Row + Column + units,
                      residual = ~ ar1(Row):ar1(Column),
                      data=Wheat.dat)

# Load current fit into an asrtests object
current.asrt <- as.asrtests(current.asr, NULL, NULL)

# check the class of current.asrt
is.asrtests(current.asrt)

## End(Not run)
```

---

is.predictions.frame *Tests whether an object is of class predictions.frame*

Description

A single-line function that tests whether an object is of class predictions.frame.

Usage

```
is.predictions.frame(object)
```

Arguments

object	An object to be tested.

Value

A logical.

Author(s)

Chris Brien

See Also

asremlPlus-package, predictions.frame, validPredictionsFrame, as.predictions.frame

Examples

```
  data(Oats.dat)

  ## Use asreml to get predictions and associated statistics

  ## Not run:
  m1.asr <- asreml(Yield ~ Nitrogen*Variety,
                   random=~Blocks/Wplots,
                   data=Oats.dat)
  current.asrt <- as.asrtests(m1.asr)
  Var.pred <- asreml::predict.asreml(m1.asr, classify="Nitrogen:Variety",
                                     sed=TRUE)
  if (getASRemlVersionLoaded(nchar = 1) == "3")
    Var.pred <- Var.pred$predictions
  Var.preds <- as.predictions.frame(Var.pred$pvals, se = "std.error",
                                    est.status = "status")

## End(Not run)

  ## Use lmerTest and emmmeans to get predictions and associated statistics
  if (requireNamespace("lmerTest", quietly = TRUE) &
      requireNamespace("emmeans", quietly = TRUE))
  {
    m1.lmer <- lmerTest::lmer(Yield ~ Nitrogen*Variety + (1|Blocks/Wplots),
                              data=Oats.dat)
    Var.emm <- emmeans::emmeans(m1.lmer, specs = ~ Nitrogen:Variety)
    Var.preds <- summary(Var.emm)
    Var.preds <- as.predictions.frame(Var.preds, predictions = "emmean",
                                      se = "SE", interval.type = "CI",
                                      interval.names = c("lower.CL", "upper.CL"))
  }

  if (exists("Var.preds"))
  {
    ## Check the class and validity of the alldiffs object
    is.predictions.frame(Var.preds)
  }
```

---

isCompoundSymmetric.matrix

Tests whether an object of class matrix is compound symmetric

---

Description

Tests whether an object of class matrix is compound symmetric by checking whether all diagonal elements are equal and all off-diagonal elements are equal.

Usage

```
## S3 method for class 'matrix'
isCompoundSymmetric(object, tol = 100 * .Machine$double.eps, ...)
```

Arguments

object	An object to be tested.
tol	a numeric scalar > 0 specifying that values smaller than it are considered to be zero.
...	Provision for passing arguments to functions called internally - not used at present.

Value

A logical.

Author(s)

Chris Brien

See Also

isSymmetric

Examples

```
  data(Oats.dat)

  ## Not run:
  ## Use asreml to get the variance matrix of a set of predictions

  m1.asr <- asreml(Yield ~ Nitrogen*Variety,
                   random=~Blocks/Wplots,
                   data=Oats.dat)
  current.asrt <- as.asrtests(m1.asr)
  Var.pred <- asreml::predict.asreml(m1.asr, classify="Nitrogen:Variety",
                                     vcov = TRUE)
                                    est.status = "status")
  isCompoundSymmetrix(Var.pred$vcov)

## End(Not run)

  ## Use lmerTest and emmmeans to get the variance matrix of a set of predictions
  if (requireNamespace("lmerTest", quietly = TRUE) &
      requireNamespace("emmeans", quietly = TRUE))
  {
    m1.lmer <- lmerTest::lmer(Yield ~ Nitrogen*Variety + (1|Blocks/Wplots),
                              data=Oats.dat)
    Var.emm <- emmeans::emmeans(m1.lmer, specs = ~ Nitrogen:Variety)
    Var.vcov <- vcov(Var.emm)
```

```
}

if (exists("Var.vcov"))
{
  ## Check the variance matrix of the predictions is compound symmetric
  isCompoundSymmetric(Var.vcov)
}
```

`iterate.asrtests` *Subject the fitted* `asreml.obj` *stored in an* `asrtests.object` *to further iterations of the fitting process.*

Description

In an effort to improve convergence, subject the fitted `asreml.obj` stored in an `asrtests.object` to further iterations of the fitting process; the model specification is not changed. While no change is made to the `test.summary`, the `wald.tab` is updated.

Usage

```
## S3 method for class 'asrtests'
iterate(asrtests.obj, denDF="numeric", trace = FALSE, ...)
```

Arguments

`asrtests.obj`	an `asrtests.object` containing the components (i) `asreml.obj`, (ii) `wald.tab`, and (iii) `test.summary`.
`denDF`	Specifies the method to use in computing approximate denominator degrees of freedom when `wald.asreml` is called. Can be `none` to suppress the computations, `numeric` for numerical methods, `algebraic` for algebraic methods or `default`, the default, to automatically choose numeric or algebraic computations depending on problem size. The denominator degrees of freedom are calculated according to Kenward and Roger (1997) for fixed terms in the dense part of the model.
`trace`	If TRUE then partial iteration details are displayed when ASReml-R functions are invoked; if FALSE then no output is displayed.
`...`	further arguments passed to `update.asreml`.

Value

An object of S3-class `asrtests`.

Author(s)

Chris Brien

References

Kenward, M. G., & Roger, J. H. (1997). Small sample inference for fixed effects from restricted maximum likelihood. *Biometrics*, **53**, 983-997.

See Also

`as.asrtests`, `asrtests.object`, `newfit.asreml`

Examples

```
## Not run:
  current.asrt <- iterate(current.asrt)

## End(Not run)
```

`Ladybird.dat` *Data for an experiment to investigate whether ladybirds transfer aphids*

Description

Welham et al. (2015, Example 8.2) describe a three-factor factorial experiment to investigate whether ladybirds transfer fungus to live aphids on plants. The three factors are Host plant (beans, trefoil), infected Cadavers (5, 10, 20), and Ladybird (-, +). A generalized randomized complete-block design is used to assign the three factors to 2 Runs, each of which involves 36 containers with a plant and live aphids. The response to be analyzed is the logit of the proportion of live aphids that were infected.

The columns in the data frame are: ID, Run, Plant, Host, Ladybird, Cadavers, Live, Infected, logitP, Prop. The column ID numbers the observations. Live, Infected, logitP, Prop are response variables.

Usage

```
data(Ladybird.dat)
```

Format

A data.frame containing 72 observations of 10 variables.

Author(s)

Chris Brien

Source

Welham, S. J., Gezan, S. A., Clark, S. J., & Mead, A. (2015). *Statistical Methods in Biology: Design and Analysis of Experiments and Regression*. Boca Raton: Chapman and Hall/CRC..

linTransform.alldiffs *Calculates a linear transformation of the predictions stored in an* alldiffs.object.

Description

Effects the linear transformation of the predictions in the supplied alldiffs.object, the transformation being specified by a matrix or a formula. The values of the transformed values are stored in an alldiffs.object. A matrix might be a contrast matrix or a matrix of weights for the levels of a factor used to obtain the weighted average over the levels of that factor. A formula gives rise to a projection matrix that linearly transforms the predictions so that they conform to the model specified by the formula, this model being a submodel of that inherent in the classify.

If pairwise = TRUE, all pairwise differences between the linear transforms of the predictions, their standard errors, p-values and LSD statistics are computed as using allDifferences.data.frame. This adds them to the alldiffs.object as additional list components named differences, sed, p.differences and LSD.

If a transformation has been applied (any one of transform.power is not one, scale is not one and offset is nonzero), the backtransforms of the transformed values and of the lower and upper limits of their error.intervals are added to a data.frame that is consistent with a predictions.frame. If transform.power is other than one, the standard.error column of the data.frame is set to NA. This data.frame is added to the alldiffs.object as a list component called backtransforms.

The printing of the components produced is controlled by the tables argument. The order of plotting the levels of one of the factors indexing the predictions can be modified and is achieved using sort.alldiffs.

Usage

```
## S3 method for class 'alldiffs'
linTransform(alldiffs.obj, classify = NULL, term = NULL,
             linear.transformation = NULL, EGLS.linTransform = TRUE,
             Vmatrix = FALSE, error.intervals = "Confidence",
             avsed.tolerance = 0.25, accuracy.threshold = NA,
             LSDtype = "overall", LSDsupplied = NULL,
             LSDby = NULL, LSDstatistic = "mean",
             LSDaccuracy = "maxAbsDeviation",
             zero.tolerance = .Machine$double.eps ^ 0.5,
             response = NULL, response.title = NULL,
             x.num = NULL, x.fac = NULL,
             tables = "all", level.length = NA,
             pairwise = TRUE, alpha = 0.05,
             inestimable.rm = TRUE, ...)
```

Arguments

alldiffs.obj An alldiffs.object.

`classify` A character string giving the variables that define the margins of the multiway table corresponding to the `predictions` in `alldiffs.obj`. Multiway tables are specified by forming an interaction type term from the classifying variables, that is, separating the variable names with the : operator.

`term` A character string giving the variables that define the term that was fitted using `asreml` and that corresponds to `classify`. It only needs to be specified when it is different to `classify`; it is stored as an attribute of the `alldiffs.object`. It is likely to be needed when the fitted model includes terms that involve both a `numeric` covariate and a `factor` that parallel each other; the `classify` would include the covariate and the `term` would include the `factor`.

`linear.transformation`

A `formula` or a `matrix`. If a `formula` is given then it is taken to be a submodel of a model term corresponding to the `classify`. The projection matrix that transforms the `predictions` so that they conform to the submodel is obtained; the submodel does not have to involve variables in the `classify`, but the variables must be columns in the `predictions` component of `alldiffs.obj` and the space for the submodel must be a subspace of the space for the term specified by the `classify`. For example, for `classify` set to `"A:B"`, the submodel `~ A + B` will result in the `predictions` for the combinations of `A` and `B` being made additive for the `factor`s `A` and `B`. The submodel space corresponding to `A + B` is a subspace of the space `A:B`. In this case both the submodel and the classify involve only the factors A and B. To fit an intercept-only submodel, specify `linear.transformation` to be the formula `~1`.

If a `matrix` is provided then it will be used to apply the linear transformation to the `predictions`. The number of rows in the `matrix` should equal the number of linear combinations of the `predictions` desired and the number of columns should equal the number of `predictions`.

In either case, as well as the values of the linear combinations, their standard errors, pairwise differences and associated statistics are returned.

`EGLS.linTransform`

A `logical` indicating whether or not the `linear.transformation` of the predictions stored in an `alldiffs.object` by fitting a submodel supplied in a `formula` is to take into account the variance of the predictions using a Estimated Generalized Least Squares (EGLS) approach. This is likely to be appropriate when the variance matrix of the predictions is not compound symmetric i.e. when not all the variances are equal or not all the covariances are equal. If the variance matrix is compund symmetric, then the setting of `EGLS.linTransform` will not affect the transformed predictions.

`Vmatrix` A `logical` indicating whether the variance matrix of the `predictions` will be stored as a component of the `alldiffs.object` that is returned. If `linear.transformation` is set, it will be stored irrespective of the value of `Vmatrix`.

`error.intervals`

A `character` string indicating the type of error interval, if any, to calculate in order to indicate uncertainty in the results. Possible values are `"none"`, `"StandardError"`, `"Confidence"` and `"halfLeastSignificant"`. The default is for confidence limits to be used. The `"halfLeastSignificant"` option results in half the Least Significant Difference (LSD) being added and subtracted

to the predictions, the LSD being calculated using the square root of the mean of the variances of all or a subset of pairwise differences between the predictions. If the LSD is zero, as can happen when predictions are constrained to be equal, then the limits of the error intervals are set to NA. If LSDtype is set to overall, the avsed.tolerance is not NA and the range of the SEDs divided by the average of the SEDs exceeds avsed.tolerance then the error.intervals calculations and the plotting will revert to confidence intervals.

avsed.tolerance

A numeric giving the value of the SED range, the range of the SEDs divided by the square root of the mean of the variances of all or a subset of the pairwise differences, that is considered reasonable in calculating error.intervals. To have it ignored, set it to NA. It should be a value between 0 and 1. The following rules apply:

1. If avsed.tolerance is NA then mean LSDs of the type specified by LSDtype are calculated and used in error.intervals and plots.
2. Irrespective of the setting of LSDtype, if avsed.tolerance is not exceeded then the mean LSDs are used in error.intervals and plots.
3. If LSDtype is set to overall, avsed.tolerance is not NA, and avsed.tolerance is exceeded then error.intervals and plotting revert to confidence intervals.
4. If LSDtype is set to factor.combinations and avsed.tolerance is not exceeded for any factor combination then the half LSDs are used in error.intervals and plots; otherwise, error.intervals and plotting revert to confidence intervals.
5. If LSDtype is set to per.prediction and avsed.tolerance is not exceeded for any prediction then the half LSDs are used in error.intervals and plots; otherwise, error.intervals and plotting revert to confidence intervals.

accuracy.threshold

A numeric specifying the value of the LSD accuracy measure, which measure is specified by LSDaccuracy, as a threshold value in determining whether the hallfLeastSignificant error.interval for a predicted value is a reasonable approximation; this will be the case if the LSDs across all pairwise comparisons for which the interval's LSD was computed, as specified by LSDtype and LSDby, are similar enough to the interval's LSD, as measured by LSDaccuracy. If it is NA, it will be ignored. If it is not NA, a column of logicals named LSDwarning will be added to the predictions component of the alldiffs.object. The value of LSDwarning for a predicted.value will be TRUE if the value of the LSDaccuracy measure computed from the LSDs for differences between this predicted.value and the other predicted.values as compared to its assignedLSD exceeds the value of accuracy.threshold. Otherwise, the value of LSDwarning for a predicted.value will be FALSE.

LSDtype

A character string that can be overall, factor.combinations, per.prediction or supplied. It determines whether the values stored in a row of a LSD.frame are the values calculated (i) overall from the LSD values for all pairwise comparisons, unless there is only one, possibly repeated, prediction, when a notional LSD is calculated, (ii) the values calculated from the pairwise LSDs for

the levels of each factor.combination, unless there is only one prediction for a level of the factor.combination, when a notional LSD is calculated, (iii) per.prediction, being based, for each prediction, on all pairwise differences involving that prediction, or (iv) as supplied values of the LSD, specified with the LSDsupplied argument; these supplied values are to be placed in the assignedLSD column of the LSD.frame stored in an alldiffs.object so that they can be used in LSD calculations.

See LSD.frame for further information on the values in a row of this data.frame and how they are calculated.

LSDsupplied A data.frame or a named numeric containing a set of LSD values that correspond to the observed combinations of the values of the LSDby variables in the predictions.frame or a single LSD value that is an overall LSD. If a data.frame, it may have (i) a column for the LSDby variable and a column of LSD values or (ii) a single column of LSD values with rownames being the combinations of the observed values of the LSDby variables. Any name can be used for the column of LSD values; assignedLSD is sensible, but not obligatory. Otherwise, a numeric containing the LSD values, each of which is named for the observed combination of the values of the LSDby variables to which it corresponds. (Applying the function dae::fac.combine to the predictions component is one way of forming the required combinations for the (row) names.) The values supplied will be incorporated into assignedLSD column of the LSD.frame stored as the LSD component of the alldiffs.object.

LSDby A character (vector) of variables names, being the names of the factors or numerics in the classify; for each combination of their levels and values, there will be or is a row in the LSD.frame stored in the LSD component of the alldiffs.object when LSDtype is factor.combinatons.

LSDstatistic A character nominating one or more of minimum, q10, q25, mean, median, q75, q90 or maximum as the value(s) to be stored in the assignedLSD column in an LSD.frame; the values in the assignedLSD column are used in computing halfLeastSignificant error.intervals. Here q10, q25, q75 and q90 indicate the sample quantiles corresponding to probabilities of 0.1, 0.25, 0.75 and 0.9 for the group of LSDs from which a single LSD value is calculated. The function quantile is used to obtain them. The mean LSD is calculated as the square root of the mean of the squares of the LSDs for the group. The median is calculated using the median function. Multiple values are only produced for LSDtype set to factor.combination, in which case LSDby must not be NULL and the number of values must equal the number of observed combinations of the values of the variables specified by LSDby. If LSDstatistic is NULL, it is reset to mean.

LSDaccuracy A character nominating one of maxAbsDeviation, maxDeviation, q90Deviation or RootMeanSqDeviation as the statistic to be calculated as a measure of the accuracy of assignedLSD. The option q90Deviation produces the sample quantile corresponding to a probability of 0.90. The deviations are the differences between the LSDs used in calculating the LSD statistics and each assigned LSD and the accuracy is expressed as a proportion of the assigned LSD value. The calculated values are stored in the column named accuracyLSD in an LSD.frame.

zero.tolerance A numeric specifying the value such that if a predicted.value, its variance-

covariance, or an LSD is less than it, the LSD will be considered to be zero.

response — A character specifying the response variable for the predictions. It is stored as an attribute to the alldiffs.object .

response.title — A character specifying the title for the response variable for the predictions. It is stored as an attribute to the alldiffs.object.

x.num — A character string giving the name of the numeric covariate that (i) is potentially included in terms in the fitted model and (ii) is the x-axis variable for plots. Its values will not be converted to a factor.

x.fac — A character string giving the name of the factor that (i) corresponds to x.num and (ii) is potentially included in terms in the fitted model. It should have the same number of levels as the number of unique values in x.num. The levels of x.fac must be in the order in which they are to be plotted - if they are dates, then they should be in the form yyyymmdd, which can be achieved using as.Date. However, the levels can be non-numeric in nature, provided that x.num is also set.

tables — A character vector containing a combination of none, predictions, vcov, backtransforms, differences, p.differences, sed, LSD and all. These nominate which components of the alldiffs.object to print.

level.length — The maximum number of characters from the levels of factors to use in the row and column labels of the tables of pairwise differences and their p-values and standard errors.

pairwise — A logical indicating whether all pairwise differences of the predictions and their standard errors and p-values are to be computed and stored. If tables is equal to "differences" or "all" or error.intervals is equal to "halfLeastSignificant", they will be stored irrespective of the value of pairwise.

alpha — A numeric giving the significance level for LSDs or one minus the confidence level for confidence intervals. It is stored as an attribute to the alldiffs.object.

inestimable.rm — A logical indicating whether rows for predictions that are not estimable are to be removed from the components of the alldiffs.object.

... — further arguments passed to redoErrorIntervals.alldiffs.

Details

For a matrix $\mathbf{L}$, vector of predictions $\mathbf{p}$ and variance matrix of the predictions $\mathbf{V}_p$, the linear transformed predictions are given by $\mathbf{Lp}$ with variance matrix $\mathbf{L}\mathbf{V}_p\mathbf{L}^{\mathrm{T}}$. The last matrix is used to compute the variance of pairwise differences between the transformed values.

If linear.transformation is a matrix, $\mathbf{M}$ say, then the linear-transformation matrix, $\mathbf{L}$, is just the supplied matrix $\mathbf{M}$.

If linear.transformation is a formula and EGLS.linTransform is TRUE, then a matrix $\mathbf{M}$ is obtained that is the design matrix for all of the terms in the formula. Using $\mathbf{M}$, the linear-transformation matrix, $\mathbf{L}$, is formed as $\mathbf{M}(\mathbf{M}^{\top}\widehat{\mathbf{V}}^{-}\mathbf{M})^{-}(\mathbf{M}^{\top}\widehat{\mathbf{V}}^{-})$.

On the other hand, for linear.transformation a formula and EGLS.linTransform set to FALSE, $\mathbf{L}$ is formed as the sum of the orthogonal projection matrices obtained using pstructure.formula from the package dae; grandMean is set to TRUE and orthogonalize to "eigenmethods".

Value

A alldiffs.object with the linear transformation of the predictions and their standard errors and all pairwise differences between the linear transforms of their predictions, their standard errors and p-values and LSD statistics.

If the supplied alldiffs.object contained a backtransforms component, then the returned alldiffs.object will contain a backtransforms component with the backtransformed linear transformation of the predictions. The backtransformation will, after backtransforming for any power transformation, subtract the offset and then divide by the scale.

If error.intervals is not "none", then the predictions component and, if present, the backtransforms component will contain columns for the lower and upper values of the limits for the interval. The names of these columns will consist of three parts separated by full stops: 1) the first part will be lower or upper; 2) the second part will be one of Confidence, StandardError or halfLeastSignificant; 3) the third component will be limits.

The name of the response, the response.title, the term, the classify, tdf, alpha, sortFactor and the sortOrder will be set as attributes to the object. Also, if error.intervals is "halfLeastSignificant", then those of LSDtype, LSDby and LSDstatistic that are not NULL will be added as attributes of the object and of the predictions frame; additionally, LSDvalues will be added as attribute of the predictions frame, LSDvalues being the LSD values used in calculating the error.intervals.

Author(s)

Chris Brien

See Also

linTransform, predictPlus.asreml, as.alldiffs, print.alldiffs, sort.alldiffs, subset.alldiffs, allDifferences.data.frame, redoErrorIntervals.alldiffs, recalcLSD.alldiffs, pickLSDstatistics.alldiffs, predictPresent.asreml, plotPredictions.data.frame, as.Date, predict.asreml

Examples

```
data(WaterRunoff.dat)

##Use asreml to get predictions and associated statistics

## Not run:
asreml.options(keep.order = TRUE) #required for asreml-R4 only
current.asr <- asreml(fixed = pH ~ Benches + (Sources * (Type + Species)),
                      random = ~ Benches:MainPlots,
                      keep.order=TRUE, data= WaterRunoff.dat)
current.asrt <- as.asrtests(current.asr, NULL, NULL)
#Get additive predictions directly using predictPlus
diffs.sub <- predictPlus.asreml(classify = "Sources:Species", Vmatrix = TRUE,
                                linear.transformation = ~ Sources + Species,
                                asreml.obj = current.asr, tables = "none",
                                wald.tab = current.asrt$wald.tab,
                                present = c("Type","Species","Sources"))
```

```
## End(Not run)

## Use lmeTest and emmmeans to get predictions and associated statistics

if (requireNamespace("lmerTest", quietly = TRUE) &
    requireNamespace("emmeans", quietly = TRUE))
{
  m1.lmer <- lmerTest::lmer(pH ~ Benches + (Sources * Species) +
                              (1|Benches:MainPlots),
                            data=na.omit(WaterRunoff.dat))
  SS.emm <- emmeans::emmeans(m1.lmer, specs = ~ Sources:Species)
  SS.preds <- summary(SS.emm)
  den.df <- min(SS.preds$df, na.rm = TRUE)
  ## Modify SS.preds to be compatible with a predictions.frame
  SS.preds <- as.predictions.frame(SS.preds, predictions = "emmean",
                                   se = "SE", interval.type = "CI",
                                   interval.names = c("lower.CL", "upper.CL"))

  ## Form an all.diffs object and check its validity
  SS.vcov <- vcov(SS.emm)
  SS.diffs <- allDifferences(predictions = SS.preds, classify = "Sources:Species",
                             vcov = SS.vcov, tdf = den.df)
  validAlldiffs(SS.diffs)

  #Get additive predictions
  diffs.sub <- linTransform(SS.diffs, classify = "Sources:Species",
                            linear.transformation = ~ Sources + Species,
                            Vmatrix = TRUE, tables = "none")
}

##Calculate contrasts from prediction obtained using asreml or lmerTest
if (exists("diffs.sub"))
{
  #Contrast matrix for differences between each species and non-planted for the last source
  L <- cbind(matrix(rep(0,7*32), nrow = 7, ncol = 32),
             diag(1, nrow = 7),
             matrix(rep(-1, 7), ncol = 1))
  rownames(L) <- as.character(diffs.sub$predictions$Species[33:39])
  diffs.L <- linTransform(diffs.sub,
                          classify = "Sources:Species",
                          linear.transformation = L,
                          tables = "predictions")
}
```

`loadASRemlVersion` *Ensures that a specific version of* `asreml` *is loaded.*

Description

Loads the specified version of `asreml`, provided that it is not already loaded. If the version of `asreml` is not the required version, then the loaded version is unloaded first.

Usage

```
loadASRemlVersion(version = 4, ...)
```

Arguments

version	The version that is to be loaded, the version consisting of just the initial characters that are significant in the version that should be loaded. For example, the default value of 4 implies that any version that begins with "4" is acceptable. It is used to check that the required version is loaded.
...	Other library/require arguments that are needed to load the specified version of asreml.

Value

A character, being all characters in the version of asreml that is loaded on exit from the function.

Author(s)

Chris Brien

See Also

getASRemlVersionLoaded.

Examples

```
## Not run:
loadASRemlVersion(3, lib.loc = "D:\Analyses\R asreml3")
## End(Not run)
```

LSD.frame *Description of an LSD frame*

Description

A data.frame that stores Least Significant differences (LSDs) for predictions for a fitted model.

Value

A data.frame that can be a component of an alldiffs.object and that contains LSD values and statistics to be used in determining the significance of the pairwise differences. In particular, they are used in calculating halfLeastSignificant limits to be included in a predictions.frame.

Exactly what an LSD.frame contains is determined by the following arguments to functions that return an alldiffs.object: LSDtype, LSDby, LSDstatistic, LSDaccuracy and LSDsupplied. The rownames of the LSD.frame indicate, for each of its rows, for what group of predictions the entries in the row were calculated, this being controlled by the LSDtype and LSDby arguments.

The values for all of the LSD arguments are stored as attributes to the alldiffs.object and the predictions and, if present backtransforms, components of the alldiffs.object.

An LSD.frame always has the eight columns c, minimumLSD, meanLSD, maximumLSD, assignedLSD, accuracyLSD, falsePos and falseNeg.

1. c: This gives the number of pairwise comparison of predictions for the combinations of the factor levels given by the row name. If the row name is overall then it is for all predictions.
2. minimumLSD, meanLSD, maximumLSD: These are computed for either overall, factor.combinations, per.prediction or supplied LSD values, as specified by the LSDtype argument. The meanLSD is calculated using the square root of the mean of the variances of set of pairwise differences appropriate to the specific LSDtype argument.

   For overall, the mean, minimum and maximum of the LSDs for all pairwise comparisons are computed.

   If factor.combinations was specified for LSDtype when the LSDs were being calculated, then the LSD.frame contains a row for each combination of the values of the factors and numerics specified by LSDby. The values in a row are calculated from the LSD values for the pairwise differences for each combination of the factors and numerics values, unless there is only one prediction for a combination, when notional LSDs are calculated that are based on the standard error of the prediction multiplied by the square root of two. Notional LSDs indicate the LSD that would apply to the difference between the prediction and, if it existed, another prediction with the same standard error.

   For per.prediction, the minimum, mean and maximum LSD, based, for each prediction, on the LSD values for all pairwise differences involving that prediction are computed.

   For supplied, the LSD.frame is set up based on the setting of LSDby: a single row with name overall if LSDby is NULL or, if LSDby is a vector of factor and numeric names, rows for each observed combinations of the values of the named factors and numerics. The LSDsupplied argument is used to provide the values to be stored in the column assignedLSD.
3. assignedLSD: The assignedLSD column contains the values that are assigned for use in calculating halfLeastSignificant error.intervals. Its contents are determined by LSDstatistic and LSDsupplied arguments. The LSDsupplied argument allows the direct specification of values to be placed in the assignedLSD column of the LSD.frame. The default is to use the values in the meanLSD column.
4. LSDaccuracy: The LSDaccuracy gives an indication of the proportion that the correct LSD for a single predicted.value might deviate from its assignedLSD value. The contents of the accuracyLSD column is controlled by the LSDaccuracy argument.
5. falsePos and falseNeg: These columns contain the number of false positives and negatives if the assignedLSD value(s) is(are) used to determine the significance of the pairwise predictions differences. Each LSD value in the assignedLSD column is used to determine the significance of pairwise differences that involve predictions for the combination of values given by the row name for the LSD value.

See recalcLSD.alldiffs for more information.

Author(s)

Chris Brien

See Also

recalcLSD.alldiffs, redoErrorIntervals.alldiffs, predictPresent.asreml, predictPlus.asreml

Examples

```
  data(Oats.dat)

  ## Use asreml to get predictions and associated statistics

  ## Not run:
  m1.asr <- asreml(Yield ~ Nitrogen*Variety,
                   random=~Blocks/Wplots,
                   data=Oats.dat)
  current.asrt <- as.asrtests(m1.asr)
  Var.diffs <- predictPlus(m1.asr, classify="Nitrogen:Variety",
                           wald.tab = current.asrt$wald.tab,
                           tables = "none")

## End(Not run)

  ## Use lmerTest and emmmeans to get predictions and associated statistics
   if (requireNamespace("lmerTest", quietly = TRUE) &
      requireNamespace("emmeans", quietly = TRUE))
  {
    m1.lmer <- lmerTest::lmer(Yield ~ Nitrogen*Variety + (1|Blocks/Wplots),
                              data=Oats.dat)
    #Get predictions
    Var.emm <- emmeans::emmeans(m1.lmer, specs = ~ Nitrogen:Variety)
    Var.preds <- summary(Var.emm)
    ## Modify Var.preds to be compatible with a predictions.frame
    Var.preds <- as.predictions.frame(Var.preds, predictions = "emmean",
                                      se = "SE", interval.type = "CI",
                                      interval.names = c("lower.CL", "upper.CL"))
    Var.vcov <- vcov(Var.emm)
    Var.sed <- NULL

    #Set up an alldiffs object, which includes overall LSDs
    Var.diffs <- allDifferences(predictions = Var.preds, classify = "Variety:Nitrogen",
                                    sed = Var.sed, vcov = Var.vcov, tdf = 45)
  }

  if (exists("Var.diffs"))
  {
    ## Use recalcLSD to get LSDs for within Variety differences
    Var.LSD.diffs <- recalcLSD(Var.diffs,
                               LSDtype = "factor.combinations", LSDby = "Variety")
    print(Var.LSD.diffs$LSD)
  }
```

makeTPPSplineMats.data.frame

Make the spline basis matrices and data needed to fit Tensor Product P-splines.

Description

Prepares the fixed and random P-spline basis matrices, and associated information, that are needed for fitting of Tensor Product P-splines (TPPS) as described by Rodriguez-Alvarez et al. (2018). This function is called internally by addSpatialModelOnIC.asrtests, addSpatialModelOnIC.asrtests and chooseSpatialModelOnIC.asrtests when fitting TPPS models for local spatial variation. There are two methods available, controlled by asreml.option for creating and storing the basis functions. This function is most likely to be called directly when mbf has been used in creating an asreml.object and it is desired to use the object in a session subsequent to the session in which the object was created.

Usage

```
## S3 method for class 'data.frame'
makeTPPSplineMats(data, sections = NULL,
                  row.covar, col.covar,
                  nsegs = NULL, nestorder = c(1,1),
                  degree = c(3,3), difforder = c(2,2),
                  rotateX = FALSE, theta = c(0,0),
                  asreml.option = "grp", mbf.env = sys.frame(),
                  ...)
```

Arguments

data	An data.frame that holds the spline bases for a section. It is indexed by columns named col and row.
sections	A single character string that species the name of the column in the data.frame that contains the factor that identifies different sections of the data to which separate spatial models are to be fitted.
row.covar	A single character string nominating a numeric column in the data.frame that contains the values of a covariate indexing the rows of the grid.
col.covar	A single character string nominating a numeric column in the data.frame that contains the values of a covariate indexing the columns of the grid.
nsegs	A pair of numeric values giving the number of segments into which the column and row ranges are to be split, respectively, for fitting a P-spline model (TPPS) (each value specifies the number of internal knots + 1). If not specified, then (number of unique values - 1) is used in each dimension; for a grid layout with equal spacing, this gives a knot at each data value. If sections is not NULL and the grid differs between the sections, then nsegs will differ between the sections.

`nestorder` A numeric of length 2. The order of nesting for column and row dimensions, respectively, in fitting a P-spline model (TPPS). A value of 1 specifies no nesting, a value of 2 generates a spline with half the number of segments in that dimension, etc. The number of segments in each direction must be a multiple of the order of nesting.

`degree` A numeric of length 2. The degree of polynomial spline to be used for column and row dimensions respectively, in fitting a P-spline (TPPS).

`differorder` A numeric of length 2. The order of differencing for column and row dimensions, respectively, in fitting a P-spline (TPPS).

`rotateX` A logical indicating whether to rotate the eigenvectors of the penalty matrix, as described by Piepho, Boer and Williams (2022), when fitting a P-spline (TPPS). Setting `rotateX` to TRUE results in a search for an optimized rotation under a model that omits the random spline interaction terms. If `ngridangles` is set to NULL, the optimal rotation us found using an optimizer (`nloptr::bobyqa`). Otherwise, the optimal rotation is found by exploring the fit over a two-dimensional grid of rotation angle pairs. The optimization seeks to optimize the criterion nominated in `which.rotacriterion`. Rotation of the eigenvectors is only relevant for `differorder` values greater than 1 and has only been implemented for `differorder` equal to 2.

`theta` A numeric of length 2. The angle (in degrees) to be used in rotating the eignevectors of the penalty matrix of a P-spline (TPPS).

`asreml.option` A single character string specifying whether the `grp` or `mbf` methods are to be used to supply externally formed covariate matrices to `asreml` when fitting a P-spline (TPPS). Compared to the `mbf` method, the `grp` method is somewhat faster, but creates large asrtests.objects for which the time it takes to save them can exceed any gains in execution speed. The `grp` method adds columns to the data.frame containing the `data`. On the other hand, the `mbf` method adds only the fixed covariates to `data` and stores the random covariates in the environment of the internal function that calls the spline-fitting function; there are three smaller data.frames for each `section` that are not stored in the `asreml.object` resulting from the fitted model.

`mbf.env` A environment specifying the environment to which the data.frames containing the spline bases are to be assigned. If `mbf.env` is NULL, the data.frames will not be assigned.

`...` Further arguments passed to `tpsmmb` from package `TPSbits`.

Details

The objects are formed using the function `tpsmmb` from the R package `TPSbits` authored by Sue Welham (2022). This function has been extended to allow for `sections` (see below) and to allow rotation of the penalty matrix for the linear component of the interaction terms in TPPCS models (for more information about rotation see Piepho, Boer and Williams, 2022).

Each combination of a `row.covar` and a `col.covar` does not have to specify a single observation; for example, to fit a local spatial variation model to the main units of a split-unit design, each combination would correspond to a main unit and all subunits of the main unit would have the same combination.

The data for experiment can be divided sections and the spline bases and associated data will be produced for each section. If there is more than one sections, then a list is returned that has a component for each section. The component for a section contains:

Value

A list of length equal to the number of sections is produced. Each of these components is a list with 8 or 9 components. The component named data.plus, being the input data.frame to which has been added the columns required to fit the TPPS model (the data.frame stored in the data component holds only the covariates from data).

List of length 8 or 9 (according to the asreml.option).

1. data = the input data frame augmented with structures required to fit tensor product splines in asreml-R. This data frame can be used to fit the TPS model.

   Added columns:

   - TP.col, TP.row = column and row coordinates
   - TP.CxR = combined index for use with smooth x smooth term
   - TP.C.n for n=1:diff.c = X parts of column spline for use in random model (where diff.c is the order of column differencing)
   - TP.R.n for n=1:diff.r = X parts of row spline for use in random model (where diff.r is the order of row differencing)
   - TP.CR.n for n=1:(diff.c*diff.r) = interaction between the two X parts for use in fixed model. The first variate is a constant term which should be omitted from the model when the constant (1) is present. If all elements are included in the model then the constant term should be omitted,eg. y ~ -1 + TP.CR.1 + TP.CR.2 + TP.CR.3 + TP.CR.4 + other terms...
   - when asreml="grp" or "sepgrp", the spline basis functions are also added into the data frame. Column numbers for each term are given in the grp list structure.
2. mbflist = list that can be used in call to asreml (so long as Z matrix data frames extracted with right names, eg BcZ<stub>.df)
3. BcZ.df = mbf data frame mapping onto smooth part of column spline, last column (labelled TP.col) gives column index
4. BrZ.df = mbf data frame mapping onto smooth part of row spline, last column (labelled TP.row) gives row index
5. BcrZ.df = mbf data frame mapping onto smooth x smooth term, last column (labelled TP.CxR) maps onto col x row combined index
6. dim = list structure, holding dimension values relating to the model:
   (a) "diff.c" = order of differencing used in column dimension
   (b) "nbc" = number of random basis functions in column dimension
   (c) "nbcn" = number of nested random basis functions in column dimension used in smooth x smooth term
   (d) "diff.r" = order of differencing used in column dimension
   (e) "nbr" = number of random basis functions in column dimension
   (f) "nbrn" = number of nested random basis functions in column dimension used in smooth x smooth term

7. trace = list of trace values for ZGZ' for the random TPspline terms, where Z is the design matrix and G is the known diagonal variance matrix derived from eigenvalues. This can be used to rescale the spline design matrix (or equivalently variance components).
8. grp = list structure, only added for setting asreml="grp". For asreml="grp", provides column indexes for each of the 5 random components of the 2D splines in data.plus. Dimensions of the components can be derived from the values in the dim item.
9. data.plus = the input data.frame to which has been added the columns required to fit tensor product splines in asreml-R. This data.frame can be used to fit the TPS model. FOr multiple sections, this data.frame will occur in the component for each section. If asreml.option is set to mbf, then this component will have the attribute mbf.env that specifies the environment to which the data.frames containing the spline bases have been assigned.

Author(s)

Chris Brien

References

Piepho, H-P, Boer, M. P. & Williams, E. R. (2022) Two-dimensional P-spline smoothing for spatial analysis of plant breeding trials. *Biometrical Journal*, **64**, 835-857.)

Rodriguez-Alvarez, M. X., Boer, M. P., van Eeuwijk, F. A., & Eilers, P. H. C. (2018). Correcting for spatial heterogeneity in plant breeding experiments with P-splines. *Spatial Statistics*, **23**, 52-71.

Welham, S. J. (2022) TPSbits: *Creates Structures to Enable Fitting and Examination of 2D Tensor-Product Splines using ASReml-R.* Version 1.0.0.

See Also

addSpatialModel.asrtests, addSpatialModelOnIC.asrtests, chooseSpatialModelOnIC.asrtests, tpsmmb from TPSbits

Examples

```
## Not run:

data(Wheat.dat)

#Add row and column covariates
Wheat.dat <- within(Wheat.dat,
                    {
                      cColumn <- dae::as.numfac(Column)
                      cColumn <- cColumn  - mean(unique(cColumn))
                      cRow <- dae::as.numfac(Row)
                      cRow <- cRow - mean(unique(cRow))
                    })

#Set up the matrices
tps.XZmat <- makeTPPSplineMats(wheat.dat,
                               row.covar = "cRow", col.covar = "cColumn")

## End(Not run)
```

`newfit.asreml` *Refits an* `asreml` *model with changed arguments by extracting, modifying and evaluating its* `call` *- an alternate to update.asreml.*

Description

Extracts the call from the `asreml.obj` and evaluates that call, replacing supplied `asreml` arguments with changed values. If `update` is `TRUE` and `set.terms` is `NULL`, the call is evaluated using the variance parameter estimates from the `call` stored in `asreml.obj`; if some variance terms in the newly fitted model are singular (S) or bound (B), a refit of the model will be tried in which the variance parameter estimates in `asreml.obj` are not used and will become the fitted model if its boundary terms are a subset of those in the fitted model stored in `asreml.obj`. If `update` is `FALSE` or `set.terms` is not `NULL`, the previous values of the variance parameters are not used as initial values for fitting the new model; `G.param` and `R.param` are set to `NULL` or to values as specified for `set.terms`. The `...` argument can be used to pass `G.param` and/or `R.param`, provided `update` is `FALSE` and `set.terms` is `NULL`.

Irrespective of whether `set.terms` is `NULL`, variance parameter names, bounds and initial.values stored in the `setvparameters` `data.frame` located in the `call` component of the `asreml.obj` are added to any `set.terms` supplied in the current call. except for those in `setvparameters` that are specified in the `set.terms` argument. In the process of fitting the model, the `setvparameters` `data.frame` stored in the supplied `asreml.obj` is updated to include the prior and current `set.terms`. Then, just before exiting `newfit.asreml`, a search for changes in the bounds of the stored terms is made. If any have changed, an attempt is made to force the values back to their values on entry.

Usage

```
## S3 method for class 'asreml'
newfit(asreml.obj, fixed., random., sparse.,
       residual., rcov., update = TRUE, trace = FALSE,
       allow.unconverged = TRUE, allow.fixedcorrelation = TRUE,
       keep.order = TRUE,
       set.terms = NULL, ignore.suffices = TRUE,
       bounds = "P", initial.values = NA, ...)
```

Arguments

`asreml.obj`	A valid `asreml` object with a component named `call` (from a previous call to either `asreml` or `update.asreml`).
`fixed.`	A character or formula specifying changes to the fixed formula. This is a two-sided formula where "." is substituted for existing components in the fixed component of `asreml.obj$call`. If changes are specified, the fixed terms will be reordered so that single-variable terms come first, followed by two-variable terms and so on.
`random.`	A character or formula specifying changes to the random formula. This is a one-sided formula where "." is substituted for existing components in the random component of `asreml.obj$call`.

`sparse.` A character or formula specifying changes to the sparse formula. This is a one-sided formula where "." is substituted for existing components in the sparse component of `asreml.obj$call`.

`residual.` A character or formula specifying changes to the error formula, used when version 4 or later of ASReml-R is loaded. This is a one-sided formula where "." is substituted for existing components in the residual component of `asreml.obj$call`.

`rcov.` A character or formula specifying changes to the error formula, used when version 3 of ASReml-R is loaded. This is a one-sided formula where "." is substituted for existing components in the residual component of `asreml.obj$call`.

`update` A logical indicating whether to use the variance parameter estimates in fitting a new model using `newfit.asreml`. If `update` is `TRUE` and `set.terms` is `NULL`, the call stored in the `asreml.obj` is evaluated using the variance parameter estimates stored in `R.param` and `G.param`, as well as the changes specified in the arguments to `newfit.asreml`. If `FALSE` or `set.terms` is not `NULL`, then the use of previous values of the variance parameters as initial values is not enforced; `G.param` and `R.param` are set to `NULL` or to values as specified for `set.terms`.

`trace` A `logical` that control output from ASReml-R. If TRUE then partial iteration details are displayed when ASReml-R functions are invoked; if FALSE then no output is displayed.

`allow.unconverged`
A `logical` indicating whether to accept a new model even when it does not converge. If `FALSE` and the fit does not converge, the supplied `asreml.obj` is returned.

`allow.fixedcorrelation`
A `logical` indicating whether to accept a new model even when it contains correlations in the model whose values have been designated as fixed, bound or singular. If `FALSE` and the new model contains correlations whose values have not been able to be estimated, the supplied `asreml.obj` is returned. The fit in the supplied the `asreml.obj` will also be tested and a warning issued if both fixed correlations are found in it and `allow.fixedcorrelation` is `FALSE`.

`keep.order` A logical value indicating whether the terms should keep their positions. If `FALSE` the terms are reordered so that main effects come first, followed by the interactions, all second-order, all third-order and so on. Effects of a given order are kept in the order specified.

`set.terms` A character vector specifying the terms that are to have bounds and/or initial values set prior to fitting. The names must match those in the `vparameters` component of the new `asreml.obj`.

`ignore.suffices`
A logical vector specifying whether the suffices of the `asreml`-assigned names of the variance terms (i.e. the information to the right of an "!", other than "R!") is to be ignored in matching elements of `terms`. If `TRUE` for an element of `terms`, the suffices are stripped from the `asreml`-assigned names. If `FALSE` for an element of `terms`, the element must exactly match an `asreml`-assigned name for a variance term. This vector must be of length one or the same length as `terms`. If it is of length one then the same action is applied to the `asreml`-assigned suffices for all the terms in `terms`.

bounds	A character vector specifying the bounds to be applied to the terms specified in set.terms. This vector must be of length one or the same length as set.terms. If it is of length one then the same constraint is applied to all the terms in set.terms. If any of the bounds are equal to NA then they are left unchanged for those terms.
initial.values	A character vector specifying the initial values for the terms specified in terms. This vector must be of length one or the same length as terms. If it is of length one then the same initial value is applied to all the terms in terms. If any of the initial.values are equal to NA then they are left unchanged for those terms.
...	additional arguments to the call, or arguments with changed values.

Value

An asreml object.

Author(s)

Chris Brien

References

Butler, D. G., Cullis, B. R., Gilmour, A. R., Gogel, B. J. and Thompson, R. (2023). *ASReml-R Reference Manual Version 4.2.* VSN International Ltd, https://asreml.kb.vsni.co.uk/.

See Also

convAsremlobj.asreml, update.asreml, setvarianceterms.call

Examples

```
## Not run:
    m2.asreml <- newfit(m1.asreml, random. = "~ . - Blocks:Plots", maxiter=75)

## End(Not run)
```

---

num.recode *Recodes the unique values of a vector using the values in a new vector.*

Description

Recodes the unique values of a variate using the value in position i of the new.values vector to replace the ith sorted unique values of x. The new levels do not have to be unique.

Usage

```
num.recode(x, new.values)
```

Arguments

x	The vector to be recoded.
new.values	A vector of length unique(x) containing values to use in the recoding.

Value

A vector.

Author(s)

Chris Brien

See Also

dae::fac.recast.

Examples

```
## set up a factor with labels
x <- rep(c(-42, -14, 14, 42), 4)

## recode x
b <- num.recode(x, c(0, 28, 56, 84))
```

Oats.dat *Data for an experiment to investigate nitrogen response of 3 oats varieties*

Description

Yates (1937) describes a split-plot experiment that investigates the effects of three varieties of oats and four levels of Nitrogen fertilizer. The varieties are assigned to the main plots using a randomized complete block design with 6 blocks and the nitrogen levels are randomly assigned to the subplots in each main plot.

The columns in the data frame are: Blocks, Wplots, Subplots, Variety, Nitrogen, xNitrogen, Yield. The column xNitrogen is a numeric version of the factor Nitrogen. The response variable is Yield.

Usage

```
data(Oats.dat)
```

Format

A data.frame containing 72 observations of 7 variables.

Author(s)

Chris Brien

Source

Yates, F. (1937). The Design and Analysis of Factorial Experiments. *Imperial Bureau of Soil Science, Technical Communication*, **35**, 1-95.

`pairdiffsTransform.alldiffs`

Calculates the differences between nominated pairs of predictions stored in an `alldiffs.object`.

Description

Predictions of differences and their error intervals are formed for two levels of a factor, the `pairs.factor`. For each pair of a level of the `pairs.factor` in `numerator.levels` with a level in `denominator.levels`, an `alldiffs.object` is formed that contains the differences between predictions with this pair of levels for all of the combinations of the levels of the other factors in the `classify` of the `alldiffs.object`. These prediction differences are obtained using `linTransform` by forming a suitable contrast matrix to specify the `linear.transformation`. This function has the advantage that the factors indexing the differences are included in the components of the `alldiffs.object`s.

If `pairwise = TRUE`, all pairwise differences between the linear transforms of the `predictions`, their standard errors, p-values and LSD statistics are computed as using `allDifferences.data.frame`. This adds them to the `alldiffs.object` as additional `list` components named `differences`, `sed`, `p.differences` and `LSD`.

The printing of the components produced is controlled by the `tables` argument. The order of plotting the levels of one of the factors indexing the predictions can be modified and is achieved using `sort.alldiffs`.

Usage

```
## S3 method for class 'alldiffs'
pairdiffsTransform(alldiffs.obj, pairs.factor, first.levels, second.levels,
                   Vmatrix = FALSE, error.intervals = "Confidence",
                   avsed.tolerance = 0.25, accuracy.threshold = NA,
                   LSDtype = "overall", LSDsupplied = NULL, LSDby = NULL,
                   LSDstatistic = "mean", LSDaccuracy = "maxAbsDeviation",
                   response = NULL, response.title = NULL, tables = "all",
                   pairwise = TRUE, alpha = 0.05, ...)
```

Arguments

`alldiffs.obj`	An `alldiffs.object`.
`pairs.factor`	A `character` string giving the name of the factor for whose levels the differences are to be calculated.
`first.levels`	A `character` string containing the levels of the `pairs.factor` whose predictions are those subtracted from.

second.levels A character string containing the levels of the pairs.factor whose predictions are those that are subtracted.

Vmatrix A logical indicating whether the variance matrix of the predictions will be stored as a component of the alldiffs.object that is returned.

error.intervals
A character string indicating the type of error interval, if any, to calculate in order to indicate uncertainty in the results. Possible values are "none", "StandardError", "Confidence" and "halfLeastSignificant". The default is for confidence limits to be used. The "halfLeastSignificant" option results in half the Least Significant Difference (LSD) being added and subtracted to the predictions, the LSD being calculated using the square root of the mean of the variances of all or a subset of pairwise differences between the predictions. If the LSD is zero, as can happen when predictions are constrained to be equal, then the limits of the error intervals are set to NA. If LSDtype is set to overall, the avsed.tolerance is not NA and the range of the SEDs divided by the average of the SEDs exceeds avsed.tolerance then the error.intervals calculations and the plotting will revert to confidence intervals.

avsed.tolerance
A numeric giving the value of the SED range, the range of the SEDs divided by the square root of the mean of the variances of all or a subset of the pairwise differences, that is considered reasonable in calculating error.intervals. To have it ignored, set it to NA. It should be a value between 0 and 1. The following rules apply:

1. If avsed.tolerance is NA then mean LSDs of the type specified by LSDtype are calculated and used in error.intervals and plots.
2. Irrespective of the setting of LSDtype, if avsed.tolerance is not exceeded then the mean LSDs are used in error.intervals and plots.
3. If LSDtype is set to overall, avsed.tolerance is not NA, and avsed.tolerance is exceeded then error.intervals and plotting revert to confidence intervals.
4. If LSDtype is set to factor.combinations and avsed.tolerance is not exceeded for any factor combination then the half LSDs are used in error.intervals and plots; otherwise, error.intervals and plotting revert to confidence intervals.
5. If LSDtype is set to per.prediction and avsed.tolerance is not exceeded for any prediction then the half LSDs are used in error.intervals and plots; otherwise, error.intervals and plotting revert to confidence intervals.

accuracy.threshold
A numeric specifying the value of the LSD accuracy measure, which measure is specified by LSDaccuracy, as a threshold value in determining whether the hallfLeastSignificant error.interval for a predicted value is a reasonable approximation; this will be the case if the LSDs across all pairwise comparisons for which the interval's LSD was computed, as specified by LSDtype and LSDby, are similar enough to the interval's LSD, as measured by LSDaccuracy. If it is NA, it will be ignored. If it is not NA, a column of logicals named LSDwarning will be added to the predictions component of the alldiffs.object.

The value of LSDwarning for a predicted.value will be TRUE if the value of the LSDaccuracy measure computed from the LSDs for differences between this predicted.value and the other predicted.values as compared to its assignedLSD exceeds the value of accuracy.threshold. Otherwise, the value of LSDwarning for a predicted.value will be FALSE.

LSDtype — A character string that can be overall, factor.combinations, per.prediction or supplied. It determines whether the values stored in a row of a LSD.frame are the values calculated (i) overall from the LSD values for all pairwise comparisons, unless there is only one, possibly repeated, prediction, when a notional LSD is calculated, (ii) the values calculated from the pairwise LSDs for the levels of each factor.combination, unless there is only one prediction for a level of the factor.combination, when a notional LSD is calculated, (iii) per.prediction, being based, for each prediction, on all pairwise differences involving that prediction, or (iv) as supplied values of the LSD, specified with the LSDsupplied argument; these supplied values are to be placed in the assignedLSD column of the LSD.frame stored in an alldiffs.object so that they can be used in LSD calculations.

See LSD.frame for further information on the values in a row of this data.frame and how they are calculated.

LSDsupplied — A data.frame or a named numeric containing a set of LSD values that correspond to the observed combinations of the values of the LSDby variables in the predictions.frame or a single LSD value that is an overall LSD. If a data.frame, it may have (i) a column for the LSDby variable and a column of LSD values or (ii) a single column of LSD values with rownames being the combinations of the observed values of the LSDby variables. Any name can be used for the column of LSD values; assignedLSD is sensible, but not obligatory. Otherwise, a numeric containing the LSD values, each of which is named for the observed combination of the values of the LSDby variables to which it corresponds. (Applying the function dae::fac.combine to the predictions component is one way of forming the required combinations for the (row) names.) The values supplied will be incorporated into assignedLSD column of the LSD.frame stored as the LSD component of the alldiffs.object.

LSDby — A character (vector) of variables names, being the names of the factors or numerics in the classify; for each combination of their levels and values, there will be or is a row in the LSD.frame stored in the LSD component of the alldiffs.object when LSDtype is factor.combinatons.

LSDstatistic — A character nominating one or more of minimum, q10, q25, mean, median, q75, q90 or maximum as the value(s) to be stored in the assignedLSD column in an LSD.frame; the values in the assignedLSD column are used in computing halfLeastSignificant error.intervals. Here q10, q25, q75 and q90 indicate the sample quantiles corresponding to probabilities of 0.1, 0.25, 0.75 and 0.9 for the group of LSDs from which a single LSD value is calculated. The function quantile is used to obtain them. The mean LSD is calculated as the square root of the mean of the squares of the LSDs for the group. The median is calculated using the median function. Multiple values are only produced for LSDtype set to factor.combination, in which case LSDby must not be NULL and the number of values must equal the number of observed combinations of

	the values of the variables specified by LSDby. If LSDstatistic is NULL, it is reset to mean.
LSDaccuracy	A character nominating one of maxAbsDeviation, maxDeviation, q90Deviation or RootMeanSqDeviation as the statistic to be calculated as a measure of the accuracy of assignedLSD. The option q90Deviation produces the sample quantile corresponding to a probability of 0.90. The deviations are the differences between the LSDs used in calculating the LSD statistics and each assigned LSD and the accuracy is expressed as a proportion of the assigned LSD value. The calculated values are stored in the column named accuracyLSD in an LSD.frame.
response	A character specifying the response variable for the predictions. It is stored as an attribute to the alldiffs.object .
response.title	A character specifying the title for the response variable for the predictions. It is stored as an attribute to the alldiffs.object.
tables	A character vector containing a combination of none, predictions, vcov, backtransforms, differences, p.differences, sed, LSD and all. These nominate which components of the alldiffs.object to print.
pairwise	A logical indicating whether all pairwise differences of the predictions and their standard errors and p-values are to be computed and stored. If tables is equal to "differences" or "all" or error.intervals is equal to "halfLeastSignificant", they will be stored irrespective of the value of pairwise.
alpha	A numeric giving the significance level for LSDs or one minus the confidence level for confidence intervals. It is stored as an attribute to the alldiffs.object.
...	further arguments passed to linTransform.alldiffs.

Value

A list of alldiffs.objects with a component for each combination of a first.levels with a second.levels. The name of a component will be a level from first.levels combined with a level from second.levels, separated by a comma. If the predictions in the supplied alldiffs.object are based on a response that was transformed, each alldiffs.object in the list will include a backtransforms component that contains a column labelled backtransformed.predictions, along with the backtransforms of the nominated error.intervals. The predictions and backtransforms components in an alldiffs.object will be indexed by the variables in the classify of alldiffs.obj, except that the pairs.factor is omitted. If the transformation was the logarithmic transformation, these backtransformed.predictions are predicted ratios of the untransformed response.

If sortFactor attribute is set and is not the ratio.factor, the predictions and, if present, their backtransforms will be sorted using the sortOrder attribute of the alldiffs.object, and both sortFactor and sortOrder will be set as attributes to the object.

Author(s)

Chris Brien

See Also

linTransform, ratioTransform, predictPlus.asreml, as.alldiffs, print.alldiffs, sort.alldiffs, subset.alldiffs, allDifferences.data.frame,

redoErrorIntervals.alldiffs, recalcLSD.alldiffs, pickLSDstatistics.alldiffs, predictPresent.asreml, plotPredictions.data.frame, as.Date, predict.asreml

Examples

```
#### Form the differences for log(RGR) for Salinity
load(system.file("extdata", "testDiffs.rda", package = "asremlPlus", mustWork = TRUE))
#### For the ratios for Cl per WU Temperature - use backtransforms of log-predictions
Preds.ratio.ClUp <- pairdiffsTransform(diffs.ClUp,
                                       pairs.factor = "Temperature",
                                       first.levels = "Hot",
                                       second.levels = "Cool",
                                       error.intervals = "halfLeast",
                                     tables = "backtransforms") #Backtransforms are ratios

#### Form the differences for Nitrogen compared to no Nitrogen
data("Oats.dat")
## Not run:
m1.asr <- asreml(Yield ~ Nitrogen*Variety,
                 random=~Blocks/Wplots,
                 data=Oats.dat)
current.asrt <- as.asrtests(m1.asr)
wald.tab <-  current.asrt$wald.tab
Var.diffs <- predictPlus(m1.asr, classify="Nitrogen:Variety", pairwise = TRUE,
                         Vmatrix = TRUE, error.intervals = "halfLeast",
                         LSDtype = "factor", LSDby = "Variety",
                         wald.tab = wald.tab)

## End(Not run)

 ## Use lme4 and emmmeans to get predictions and associated statistics
if (requireNamespace("lmerTest", quietly = TRUE) &
    requireNamespace("emmeans", quietly = TRUE))
{
  m1.lmer <- lmerTest::lmer(Yield ~ Nitrogen*Variety + (1|Blocks/Wplots),
                            data=Oats.dat)
  ## Set up a wald.tab
  int <- as.data.frame(rbind(rep(NA,4)))
  rownames(int) <- "(Intercept)"
  wald.tab <- anova(m1.lmer, ddf = "Kenward", type = 1)[,3:6]
  names(wald.tab) <- names(int) <- c("Df", "denDF", "F.inc", "Pr")
  wald.tab <- rbind(int, wald.tab)
  #Get predictions
  Var.emm <- emmeans::emmeans(m1.lmer, specs = ~ Nitrogen:Variety)
  Var.preds <- summary(Var.emm)
  ## Modify Var.preds to be compatible with a predictions.frame
  Var.preds <- as.predictions.frame(Var.preds, predictions = "emmean",
                                    se = "SE", interval.type = "CI",
                                    interval.names = c("lower.CL", "upper.CL"))
  Var.vcov <- vcov(Var.emm)
  Var.sed <- NULL
  den.df <- wald.tab[match("Variety", rownames(wald.tab)), "denDF"]
```

```
  #Create alldiffs object
  Var.diffs <- as.alldiffs(predictions = Var.preds,
                           sed = Var.sed, vcov = Var.vcov,
                         classify = "Nitrogen:Variety", response = "Yield", tdf = den.df)
}

if (exists("Var.diffs"))
  Preds.diffs.OatsN <- pairdiffsTransform(alldiffs.obj = Var.diffs,
                                          pairs.factor = "Nitrogen",
                                          first.levels = c("0.2","0.4","0.6"),
                                       second.levels = "0", error.intervals = "halfLeast",
                                          tables = "none")
```

`permute.square` *Permutes the rows and columns of a square matrix.*

Description

Permutes the rows and columns of a square matrix.

Usage

```
permute.square(x, permutation)
```

Arguments

`x`	A square matrix.
`permutation`	A vector specifying the new order of rows and columns.

Value

A square `matrix`.

Author(s)

Chris Brien

See Also

`permute.to.zero.lowertri`

Examples

```
  terms.marginality <-  matrix(c(1,0,0,0,0,  0,1,0,0,0, 0,1,1,0,0,
                                 1,1,1,1,0, 1,1,1,1,1), nrow=5)
  permtn <- c(1,3,2,4,5)
  terms.marginality <- permute.square(terms.marginality, permtn)
```

`permute.to.zero.lowertri` *Permutes a square matrix until all the lower triangular elements are zero.*

Description

Permutes a square matrix until all the lower triangular elements are zero.

Usage

```
permute.to.zero.lowertri(x)
```

Arguments

x	A square matrix of order n with at least n*(n-1)/2 zero elements.

Value

A square `matrix`.

Author(s)

Chris Brien

See Also

`permute.square`

Examples

```
terms.marginality <-  matrix(c(1,0,0,0,0,  0,1,0,0,0, 0,1,1,0,0,
                               1,1,1,1,0, 1,1,1,1,1), nrow=5)
terms.marginality <- permute.to.zero.lowertri(terms.marginality)
```

`pickLSDstatistics.alldiffs` *Pick LSDstatistics whose values minimize the number of errors in pairwise comparisons of predictions.*

Description

Given an alldiffs.object with an sed component, exploreLSDs.alldiffs is used to calculate the LSD values for each set of prediction comparisons specified by LSDtype and LSDby using each of the statistics minimum, q10, q25, mean, median, q75, q90 and maximum. Then the numbers of false positives and false negatives resulting from employing each of the calculated LSDs is obtained. For each set of comparisons, the LSD value(s) with the lowest number of false positives are identified and, from these, the smallest value with the lowest number of false negatives. That is, a conservative approach is taken to picking LSD values by putting the priority on avoiding false positives. Before using the LSDstatistics that this function suggests, the number of false positives and negatives generated by them should be checked. For example, it may be that there are too many false negatives and a better balance between the numbers of false positives and negatives can be identified using exploreLSDs.alldiffs,

Usage

```
## S3 method for class 'alldiffs'
pickLSDstatistics(alldiffs.obj,
                  LSDtype = "overall", LSDby = NULL,
                  alpha = 0.05, digits = 3,
                  false.pos.wt = NULL, retain.zeroLSDs = FALSE,
                  zero.tolerance = .Machine$double.eps ^ 0.5,
                  ...)
```

Arguments

alldiffs.obj	An alldiffs.object.
LSDtype	A character string that can be overall or factor.combinations. It determines whether the LSD values that are investigated are the overall minimum, quantile10, quantile25, mean, median, quantile75, quantile90, or maximum of (i) all pairwise LSDs, or (ii) the pairwise LSDs for each combination of the values of the factors and numerics named in LSDby, unless there is only one prediction for a combination, when notional LSDs are calculated. The LSDtype specified here does not have to match that used in the creating the alldiffs.object. See LSD.frame for further information on how the LSD statistics are calculated.
LSDby	A character (vector) of variables names, being the names of some of the factors or numerics in the classify; for each combination of the values of the factors and numerics, the LSD errors are to be computed when LSDtype is factor.combinatons. The LSDby specified here does not have to match that used in the creating the alldiffs.object.
alpha	A numeric specifying the significance level for an LSD to compare a pair of predictions.
digits	A numeric specifying the number of significant digits to retain in rounding the LSDs before determining the distinct rounded LSDs.
false.pos.wt	When it is not NULL, it should be a numeric that specifies the weight (e.g. 3 so that a false positive is considered to be equivalent to three false negatives) to apply to the number of false positives in calculating the weighted sums of the num-

bers of false positives and negatives to use in comparing different LSD statistics, one being the weight for the number false negatives. The `LSDstatistic` that is chosen for making comparisons will be the one that minimizes the weighted sum, has the smallest number of false positives and, amongst these, has the smallest LSD value. If it is `NULL`, the `LSDstatistic` that will be chosen is the one that minimizes the number of false negatives from amongst those that minimize the number of false positives.

If, amongst the LSD statistices, the least number of false negatives that occurs is m, then for a particular statistic with p the number of false positives, n the number of false negatives and w the value of `false.pos.wt`, that statistic will be a candidate LSD value if $(wp) + n < m$ and i.e. if $p < (m - n)/w$.

`retain.zeroLSDs`
: A `logical` indicating whether to retain or omit LSDs that are zero when calculating the summaries of LSDs.

`zero.tolerance`
: A `numeric` specifying the value such that if an LSD is less than it, the LSD will be considered to be zero.

`...`
: Provision for passing arguments to functions called internally - not used at present.

Value

A `character` of length one for `LSDby` set to `overall` or of length equal to the number of observed combinations of the values of the `factor`s and `numeric`s in `LSDby`. Each element of the returned `character` is one of `minimum`, `q10`, `q25`, `mean`, `median`, `q75`, `q90` or `maximum`, reflecting the value(s) of the LSD from amongst those calculated that minimizes the number of false positives; if there is more than one such value, then the element will be correspond to the value of the LSD from amongst those with the minimum number of false positives that minimizes the number of false negatives.

Author(s)

Chris Brien

See Also

`asremlPlus-package`, `exploreLSDs.alldiffs` `plotLSDs.data.frame`, `plotLSDs.alldiffs`, `plotLSDerrors.alldiffs`, `plotLSDerrors.data.frame`, `recalcLSD.alldiffs`, `redoErrorIntervals.alldiffs`

Examples

```
data(WaterRunoff.dat)

##Use asreml to get predictions and associated statistics

## Not run:
asreml.options(keep.order = TRUE) #required for asreml-R4 only
current.asr <- asreml(fixed = pH ~ Benches + (Sources * (Type + Species)),
                      random = ~ Benches:MainPlots,
                      keep.order=TRUE, data= WaterRunoff.dat)
```

```
current.asrt <- as.asrtests(current.asr, NULL, NULL)
TS.diffs <- predictPlus(classify = "Sources:Type",
                        asreml.obj = current.asr,
                        wald.tab = current.asrt$wald.tab,
                        present = c("Sources", "Type", "Species"))

## End(Not run)

## Use lmeTest and emmmeans to get predictions and associated statistics

if (requireNamespace("lmerTest", quietly = TRUE) &
    requireNamespace("emmeans", quietly = TRUE))
{
  m1.lmer <- lmerTest::lmer(pH ~ Benches + (Sources * (Type + Species)) +
                              (1|Benches:MainPlots),
                            data=na.omit(WaterRunoff.dat))
  TS.emm <- emmeans::emmeans(m1.lmer, specs = ~ Sources:Type)
  TS.preds <- summary(TS.emm)
  den.df <- min(TS.preds$df, na.rm = TRUE)
  ## Modify TS.preds to be compatible with a predictions.frame
  TS.preds <- as.predictions.frame(TS.preds, predictions = "emmean",
                                   se = "SE", interval.type = "CI",
                                   interval.names = c("lower.CL", "upper.CL"))

  ## Form an all.diffs object and check its validity
  TS.vcov <- vcov(TS.emm)
  TS.diffs <- allDifferences(predictions = TS.preds, classify = "Sources:Type",
                             vcov = TS.vcov, tdf = den.df)
  validAlldiffs(TS.diffs)
}

## Pick LSD statistics for calculating LSD values for pairwise comparisons of
##  the predictions obtained using asreml or lmerTest
if (exists("TS.diffs"))
{
  ##Pick the LSD values for predictions obtained using asreml or lmerTest
  LSDstat <- pickLSDstatistics(TS.diffs)
  TS.diffs <- redoErrorIntervals(TS.diffs, LSDstatistic = LSDstat)
  TS.diffs$LSD
  LSDstat <- pickLSDstatistics(TS.diffs, LSDtype = "factor.combinations",
                         LSDby = "Sources")
  TS.diffs <- redoErrorIntervals(TS.diffs, LSDtype = "factor.combinations",
                                 LSDby = "Sources", LSDstatistic = LSDstat)
  TS.diffs$LSD
}
```

---

plotLSDerrors.alldiffs

Plots a map of the errors that occur in using the computed LSD values for pairwise differences between predictions.

Description

Produces a plot of the errors that occur in using the computed LSD values for pairwise differences predictions by comparing the result obtained from using the LSDs stored in the assignedLSD column of the LSD component of the alldiffs.object with those computed from the sed component using the t-value for the df stored in the tdf attribute of the alldiffs.object. The sed component is generally a matrix whose rows and columns are labelled by the levels of one or more factors, the set of labels being the same for rows and columns. The sections argument allows multiple plots to be produced, one for each combination of the levels of the factors listed in sections. Otherwise, a single plot is produced for all observed combinations of the levels of the factors in the classify attribute for the alldiffs.object. The plots are produced using plotLSDerrors.data.frame. The order of plotting the levels of one of the factors indexing the predictions can be modified using sort.alldiffs.

Usage

```
plotLSDerrors(object, ...)
## S3 method for class 'alldiffs'
plotLSDerrors(object, alpha = 0.05, useIntervals = FALSE,
              sections = NULL, gridspacing = 0, factors.per.grid = 0,
              triangles = "both", title = NULL,
              axis.labels = TRUE, axis.text.size = 12,
              sep=",", colours = c("white","blue","red","grey"),
              ggplotFuncs = NULL, printPlot = TRUE,
              sortFactor = NULL, sortParallelToCombo = NULL,
              sortNestingFactor = NULL, sortOrder = NULL,
              decreasing = FALSE, ...)
```

Arguments

object	An alldiffs.object with both LSD and sed components that are not NULL.
alpha	A numeric giving the significance level for the LSD.
useIntervals	A logical indicating whether to use the interval limits stored in the predictions component of object, instead of the LSDs stored in the LSD component, for determining whether pairs of predictions are significantly different. It allows a check of how the error.intervals in the predictions component will perform if they are used for all pairwise predictions comparisons, whereas the comparisons to which the LSDs apply may be restricted by the setting of the LSDby attribute of object. There is no restriction on the error.intervals that can be used. However, the limits for them must be in columns in the predictions component of object and their names must end with .limits and begin with lower. and upper.
sections	A character listing the names of the factors that are to be used to break the plot into sections. A separate plot will be produced for each observed combination of the levels of these factors.
gridspacing	A numeric specifying the number(s) of rows and columns that form groups in the grid of differences. An alternative is to specify the factors.per.grid argument to have the grid spacings automatically calculated. Grids are most useful

when two or more factors index the rows and columns. If a single, nonzero number, *k* say, is given then a grid line is placed after every *k*th row and column. If a vector of values is given then the number of grid lines is the length of the vector and the spacing between each is specified by the elements of the vector.

`factors.per.grid`
: A `numeric` specifying the number of factors to include within each grid of differences. The `gridspacing` will then be computed based on the numbers of combinations observed within the levels of the remaining factors in a single plot. The `gridspacing` argument to this function will be ignored if `factors.per.grid` is greater than zero. Grids are most useful when two or more factors index the rows and columns of each plot.

`triangles`
: A `character` indicating whether the plot should include the `lower`, `upper` or `both` triangle(s).

`title`
: A `character` string giving the main title for the plot and to which is appended the levels combination of the sectioning factors, if any, for each plot.

`axis.labels`
: A `logical` indicating whether a label is to be added to the x- and y-axes. If TRUE, the label is the comma-separated list of factors whose levels combinations are involved in the prediction differences for which the LSD values are calculated.

`axis.text.size`
: A `numeric` giving the size of the `labels` on the axes of the `heatmap`.

`sep`
: A `character` giving the characters separating the levels of different factors in the row and column names of the `sed` component.

`colours`
: A vector of colours to be passed to the `ggplot` function `scale\_colour\_gradientn`.

`ggplotFuncs`
: A `list`, each element of which contains the results of evaluating a `ggplot2` function. It is created by calling the `list` function with a `ggplot2` function call for each element. It is passed to `ggplot` via `plotLSDerrors.data.frame` to be applied in creating the `ggplot` object.

`printPlot`
: A `logical` indicating whether or not the a plot is to be printed. This would be used when just the returned `data.frame` is required.

`sortFactor`
: A `character` containing the name of the `factor` that indexes the set of predicted values that determines the sorting of the components. If there is only one variable in the `classify` term then `sortFactor` can be `NULL` and the order is defined by the complete set of predicted values. If there is more than one variable in the `classify` term then `sortFactor` must be set. In this case the `sortFactor` is sorted in the same order within each combination of the values of the `sortParallelToCombo` variables: the `classify` variables, excluding the `sortFactor`. There should be only one predicted value for each unique value of `sortFactor` within each set defined by a combination of the values of the `classify` variables, excluding the `sortFactor` `factor`. The order to use is determined by either `sortParallelToCombo` or `sortOrder`.

`sortParallelToCombo`
: A `list` that specifies a combination of the values of the `factors` and `numerics`, excluding `sortFactor`, that are in `classify`. Each of the components of the supplied `list` is named for a `classify` variable and specifies a single value for it. The combination of this set of values will be used to define a subset of the

predicted values whose order will define the order of sortFactor. Each of the other combinations of the values of the factors and numerics will be sorted in parallel. If sortParallelToCombo is NULL then the first value of each classify variable, except for the sortFactor factor, in the predictions component is used to define sortParallelToCombo. If there is only one variable in the classify then sortParallelToCombo is ignored.

sortNestingFactor
: A character containing the name of the factor that defines groups of the sortFactor within which the predicted values are to be ordered. If there is only one variable in the classify then sortNestingFactor is ignored.

sortOrder
: A character vector whose length is the same as the number of levels for sortFactor in the predictions component of the alldiffs.object. It specifies the desired order of the levels in the reordered components of the alldiffs.object. The argument sortParallelToCombo is ignored.

  The following creates a sortOrder vector levs for factor f based on the values in x: levs <- levels(f)[order(x)].

decreasing
: A logical passed to order that detemines whether the order for sorting the alldiffs.object components is for increasing or decreasing magnitude of the predicted values.

...
: Provision for passsing arguments to functions called internally - not used at present.

Value

A list with components named LSDresults and plots. The LSDresults component contains the data.frame with the columns Rows, Columns, LSDresults, sections1 and sections2. This data.frame is formed using the LSD and sed components of object and is used by plotLSDerrors.data.frame in producng the plots. The plots component contains a list of ggplot objects, one for each plot produced. Multiple plots are stored in the plots component if the sections argument is set and the plots are are named for the levels combinations of the sections.

Author(s)

Chris Brien

See Also

plotLSDerrors.alldiffs, plotLSDerrors.data.frame, plotLSDs.data.frame, exploreLSDs, sort.alldiffs, subset.alldiffs, ggplot

Examples

```
##Subset WaterRunoff data to reduce time to execute
data(WaterRunoff.dat)
tmp <- subset(WaterRunoff.dat, Date == "05-18" & Benches != "3")

##Use asreml to get predictions and associated statistics

## Not run:
```

```
asreml.options(keep.order = TRUE) #required for asreml-R4 only
current.asr <- asreml(fixed = pH ~ Benches + (Sources * (Type + Species)),
                      random = ~ Benches:MainPlots,
                      keep.order=TRUE, data= tmp)
current.asrt <- as.asrtests(current.asr, NULL, NULL)
TS.diffs <- predictPlus.asreml(classify = "Sources:Type",
                               asreml.obj = current.asr, tables = "none",
                               wald.tab = current.asrt$wald.tab,
                               present = c("Type","Species","Sources"))

## End(Not run)

## Use lmeTest and emmmeans to get predictions and associated statistics

if (requireNamespace("lmerTest", quietly = TRUE) &
    requireNamespace("emmeans", quietly = TRUE))
{
  m1.lmer <- lmerTest::lmer(pH ~ Benches + (Sources * (Type + Species)) +
                              (1|Benches:MainPlots),
                            data=na.omit(WaterRunoff.dat))
  TS.emm <- emmeans::emmeans(m1.lmer, specs = ~ Sources:Type)
  TS.preds <- summary(TS.emm)
  den.df <- min(TS.preds$df, na.rm = TRUE)
  ## Modify TS.preds to be compatible with a predictions.frame
  TS.preds <- as.predictions.frame(TS.preds, predictions = "emmean",
                                   se = "SE", interval.type = "CI",
                                   interval.names = c("lower.CL", "upper.CL"))

  ## Form an all.diffs object and check its validity
  TS.vcov <- vcov(TS.emm)
  TS.diffs <- allDifferences(predictions = TS.preds, classify = "Sources:Type",
                             vcov = TS.vcov, tdf = den.df)
  validAlldiffs(TS.diffs)
}

## Plot LSD values for predictions obtained using asreml or lmerTest
if (exists("TS.diffs"))
{
  plotLSDerrors(TS.diffs, gridspacing = rep(c(3,4), c(4,2)))

  plotLSDerrors(TS.diffs, sections = "Sources", axis.labels = TRUE)

}
```

`plotLSDerrors.data.frame`

Plots a map of the supplied errors that occur in using the computed LSD values for pairwise differences between predictions.

Description

Produces a plot of the errors that have been supplied in a data.frame. The data.frame includes two factors whose levels specify, for each LSD result, which combinations of factor levels are being compared. The function plotLSDerrors.alldiffs produces such data.frames.

Usage

```
## S3 method for class 'data.frame'
plotLSDerrors(object, LSDresults = "LSDresults", x, y,
              alpha = 0.05, triangles = "both",
              gridspacing = 0, title = NULL,
              axis.labels = NULL, axis.text.size = 12,
              colours = c("white","blue","red","grey"),
              ggplotFuncs = NULL, printPlot = TRUE, ...)
```

Arguments

object	A data.frame containing the three columns specified by LSDresults, x and y.
LSDresults	A character giving the name of the column in object that contains the LSDresults values to be plotted. The column should be a character or factor with values or levels that are a subset of Ok,FN, FP and na.
x	A character giving the name of the column in object that contains the factor whose levels index the LSD values that are to be plotted in the same column.
y	A character giving the name of the column in object that contains the labels of the LSD values that are to be plotted as the rows.
alpha	A numeric giving the significance level for the LSD.
triangles	A character indicating whether the plot should include the lower, upper or both triangle(s). Here it is only used to adjust gridlines for the omission of the diagonal.
gridspacing	A numeric specifying the number(s) of rows and columns that form groups in the grid of differences. This is most useful when two or more factors index the rows and columns. If a single, nonzero number, *k* say, is given then a grid line is placed after every *k*th row and column. If a vector of values is given then the number of grid lines is the length of the vector and the spacing between each is specified by the elements of the vector.
title	A character string giving the main title for the plot.
axis.labels	A character string giving the label to use for both the x- and y-axis.
axis.text.size	A numeric giving the size of the labels on the axes of the heatmap.
colours	A vector of colours to be passed to the ggplot function scale_colour_gradientn.
ggplotFuncs	A list, each element of which contains the results of evaluating a ggplot2 function. It is created by calling the list function with a ggplot2 function call for each element. These functions are applied in creating the ggplot object.
printPlot	A logical indicating whether or not the a plot is to be printed. This would be used when just the returned ggplot object is required.
...	Provision for passing arguments to functions called internally - not used at present.

Value

An object of class "ggplot", which can be plotted using `print` or otherwise manipulated.

Author(s)

Chris Brien

See Also

`plotLSDs.data.frame`, `plotLSDs.alldiffs`, `exploreLSDs`, ggplot

Examples

```
##Subset WaterRunoff data to reduce time to execute
data(WaterRunoff.dat)
tmp <- subset(WaterRunoff.dat, Date == "05-18")

##Use asreml to get predictions and associated statistics

## Not run:
asreml.options(keep.order = TRUE) #required for asreml-R4 only
current.asr <- asreml(fixed = pH ~ Benches + (Sources * (Type + Species)),
                      random = ~ Benches:MainPlots,
                      keep.order=TRUE, data= tmp))
current.asrt <- as.asrtests(current.asr, NULL, NULL)
TS.diffs <- predictPlus.asreml(classify = "Sources:Type",
                               asreml.obj = current.asr, tables = "none",
                               wald.tab = current.asrt$wald.tab,
                               present = c("Type","Species","Sources"))

## End(Not run)

## Use lmeTest and emmmeans to get predictions and associated statistics

if (requireNamespace("lmerTest", quietly = TRUE) &
    requireNamespace("emmeans", quietly = TRUE))
{
  m1.lmer <- lmerTest::lmer(pH ~ Benches + (Sources * (Type + Species)) +
                              (1|Benches:MainPlots),
                            data=na.omit(WaterRunoff.dat))
  TS.emm <- emmeans::emmeans(m1.lmer, specs = ~ Sources:Type)
  TS.preds <- summary(TS.emm)
  den.df <- min(TS.preds$df, na.rm = TRUE)
  ## Modify TS.preds to be compatible with a predictions.frame
  TS.preds <- as.predictions.frame(TS.preds, predictions = "emmean",
                                   se = "SE", interval.type = "CI",
                                   interval.names = c("lower.CL", "upper.CL"))

  ## Form an all.diffs object and check its validity
  TS.vcov <- vcov(TS.emm)
  TS.diffs <- allDifferences(predictions = TS.preds, classify = "Sources:Type",
                             vcov = TS.vcov, tdf = den.df)
```

```
  validAlldiffs(TS.diffs)
}

## Plot LSD values for predictions obtained using asreml or lmerTest
if (exists("TS.diffs"))
{
  LSDresults <- within(reshape2::melt(TS.diffs$p.differences),
                       {
                       Var1 <- factor(Var1, levels=dimnames(TS.diffs$p.differences)[[1]])
                         Var2 <- factor(Var2, levels=levels(Var1))
                       })
  names(LSDresults) <- c("Rows","Columns","LSDresults")
  plotLSDerrors(LSDresults, x = "Rows", y = "Columns", gridspacing = rep(c(3,4), c(4,2)))
}
```

`plotLSDs.alldiffs` *Plots a heat map of computed LSD values for pairwise differences between predictions.*

Description

Produces a heat-map plot of the computed LSD values for pairwise differences between predictions by multiplying the values stored in the `sed` component of an `alldiffs` object by the `t`-value for the `df` stored in the `tdf` attribute of the object. This component is generally a matrix whose rows and columns are labelled by the levels of one or more factors, the set of labels being the same for rows and columns. The `sections` argument allows multiple plots to be produced, one for each combination of the levels of the factors listed in `sections`. Otherwise, a single plot is produced for all observed combinations of the levels of the factors in the `classify` attribute for the `alldiffs.object`. The plots are produced using `plotLSDs.data.frame`. The order of plotting the levels of one of the factors indexing the predictions can be modified using `sort.alldiffs`.

Usage

```
plotLSDs(object, ...)
## S3 method for class 'alldiffs'
plotLSDs(object, alpha = 0.05,
         sections = NULL, gridspacing = 0, factors.per.grid = 0,
         triangles = "both",
         title = NULL, axis.labels = TRUE, axis.text.size = 12,
         sep=",", colours = RColorBrewer::brewer.pal(3, "Set2"),
         ggplotFuncs = NULL, printPlot = TRUE,
         sortFactor = NULL, sortParallelToCombo = NULL,
         sortNestingFactor = NULL, sortOrder = NULL,
         decreasing = FALSE, ...)
```

Arguments

`object`	An `alldiffs.object` with an `sed` component that is not `NULL`.

alpha — A numeric giving the significance level for the LSD.

sections — A character listing the names of the factors that are to be used to break the plot into sections. A separate plot will be produced for each observed combination of the levels of these factors.

gridspacing — A numeric specifying the number(s) of rows and columns that form groups in the grid of differences. An alternative is to specify the factors.per.grid argument to have the grid spacings automatically calculated. Grids are most useful when two or more factors index the rows and columns. If a single, nonzero number, *k* say, is given then a grid line is placed after every *k*th row and column. If a vector of values is given then the number of grid lines is the length of the vector and the spacing between each is specified by the elements of the vector.

factors.per.grid — A numeric specifying the number of factors to include within each grid of differences. The gridspacing will then be computed based on the numbers of combinations observed within the levels of the remaining factors in a single plot. The gridspacing argument to this function will be ignored if factors.per.grid is greater than zero. Grids are most useful when two or more factors index the rows and columns of each plot.

triangles — A character indicating whether the plot should include the lower, upper or both triangle(s).

title — A character string giving the main title for the plot and to which is appended the levels combination of the sectioning factors, if any, for each plot.

axis.labels — A logical indicating whether a label is to be added to the x- and y-axes. If TRUE, the label is the comma-separated list of factors whose levels combinations are involved in the prediction differences for which the LSD values are calculated.

axis.text.size — A numeric giving the size of the labels on the axes of the heatmap.

sep — A character giving the characters separating the levels of different factors in the row and column names of the sed component.

colours — A vector of colours to be passed to the ggplot function scale_colour_gradientn.

ggplotFuncs — A list, each element of which contains the results of evaluating a ggplot2 function. It is created by calling the list function with a ggplot2 function call for each element. It is passed to ggplot via plotLSDs.data.frame to be applied in creating the ggplot object.

printPlot — A logical indicating whether or not the a plot is to be printed. This would be used when just the returned data.frame is required.

sortFactor — A character containing the name of the factor that indexes the set of predicted values that determines the sorting of the components. If there is only one variable in the classify term then sortFactor can be NULL and the order is defined by the complete set of predicted values. If there is more than one variable in the classify term then sortFactor must be set. In this case the sortFactor is sorted in the same order within each combination of the values of the sortParallelToCombo variables: the classify variables, excluding the sortFactor. There should be only one predicted value for each unique value of sortFactor within each set defined by a combination of the values of the

classify variables, excluding the sortFactor factor. The order to use is determined by either sortParallelToCombo or sortOrder.

sortParallelToCombo
: A list that specifies a combination of the values of the factors and numerics, excluding sortFactor, that are in classify. Each of the components of the supplied list is named for a classify variable and specifies a single value for it. The combination of this set of values will be used to define a subset of the predicted values whose order will define the order of sortFactor. Each of the other combinations of the values of the factors and numerics will be sorted in parallel. If sortParallelToCombo is NULL then the first value of each classify variable, except for the sortFactor factor, in the predictions component is used to define sortParallelToCombo. If there is only one variable in the classify then sortParallelToCombo is ignored.

sortNestingFactor
: A character containing the name of the factor that defines groups of the sortFactor within which the predicted values are to be ordered. If there is only one variable in the classify then sortNestingFactor is ignored.

sortOrder
: A character vector whose length is the same as the number of levels for sortFactor in the predictions component of the alldiffs.object. It specifies the desired order of the levels in the reordered components of the alldiffs.object. The argument sortParallelToCombo is ignored.

  The following creates a sortOrder vector levs for factor f based on the values in x: levs <- levels(f)[order(x)].

decreasing
: A logical passed to order that detemines whether the order for sorting the alldiffs.object components is for increasing or decreasing magnitude of the predicted values.

...
: Provision for passsing arguments to functions called internally - not used at present.

Value

A list with components named LSDs and plots. The LSDs component contains the data.frame with the columns Rows, Columns, LSDs, sections1 and sections2. This data.frame is formed using the sed component of object and is used by plotLSDs.data.frame in producng the plot. The plots component contains a list of ggplot objects, one for each plot produced. Multiple plots are stored in the plots component if the sections argument is set and the plots are are named for the levels combinations of the sections.

Author(s)

Chris Brien

See Also

plotLSDs.data.frame, exploreLSDs, sort.alldiffs, subset.alldiffs, ggplot

Examples

```
##Subset WaterRunoff data to reduce time to execute
data(WaterRunoff.dat)
tmp <- subset(WaterRunoff.dat, Date == "05-18" & Benches != "3")

##Use asreml to get predictions and associated statistics

## Not run:
asreml.options(keep.order = TRUE) #required for asreml-R4 only
current.asr <- asreml(fixed = pH ~ Benches + (Sources * (Type + Species)),
                      random = ~ Benches:MainPlots,
                      keep.order=TRUE, data= tmp)
current.asrt <- as.asrtests(current.asr, NULL, NULL)
TS.diffs <- predictPlus.asreml(classify = "Sources:Type",
                               asreml.obj = current.asr, tables = "none",
                               wald.tab = current.asrt$wald.tab,
                               present = c("Type","Species","Sources"))

## End(Not run)

## Use lmeTest and emmmeans to get predictions and associated statistics

if (requireNamespace("lmerTest", quietly = TRUE) &
    requireNamespace("emmeans", quietly = TRUE))
{
  m1.lmer <- lmerTest::lmer(pH ~ Benches + (Sources * (Type + Species)) +
                              (1|Benches:MainPlots),
                            data=na.omit(WaterRunoff.dat))
  TS.emm <- emmeans::emmeans(m1.lmer, specs = ~ Sources:Type)
  TS.preds <- summary(TS.emm)
  den.df <- min(TS.preds$df, na.rm = TRUE)
  ## Modify TS.preds to be compatible with a predictions.frame
  TS.preds <- as.predictions.frame(TS.preds, predictions = "emmean",
                                   se = "SE", interval.type = "CI",
                                   interval.names = c("lower.CL", "upper.CL"))

  ## Form an all.diffs object and check its validity
  TS.vcov <- vcov(TS.emm)
  TS.diffs <- allDifferences(predictions = TS.preds, classify = "Sources:Type",
                             vcov = TS.vcov, tdf = den.df)
  validAlldiffs(TS.diffs)
}

## Plot LSD values for predictions obtained using asreml or lmerTest
if (exists("TS.diffs"))
{
  plotLSDs(TS.diffs, gridspacing = rep(c(3,4), c(4,2)))

  plotLSDs(TS.diffs, sections = "Sources", axis.labels = TRUE)

}
```

`plotLSDs.data.frame` *Plots a heat map of computed LSD-values for pairwise differences between predictions.*

Description

Produces a heat-map plot of the computed LSD values for pairwise differences between predictions that are stored in a `data.frame`. The `data.frame` includes two factors whose levels specify, for each LSD value, which combinations of factor levels are being compared.

Usage

```
## S3 method for class 'data.frame'
plotLSDs(object, LSD = "LSDs", x, y, alpha = 0.05,
         triangles = "both", gridspacing = 0,
         title = NULL, axis.labels = NULL, axis.text.size = 12,
         colours = RColorBrewer::brewer.pal(3, "Set2"),
         ggplotFuncs = NULL, printPlot = TRUE, ...)
```

Arguments

`object`	A `data.frame` containing the three columns specified by `LSD`, `x` and `y`.
`LSD`	A `character` giving the name of the column in `object` that contains the LSD values to be plotted.
`x`	A `character` giving the name of the column in `object` that contains the factor whose levels index the LSD values that are to be plotted in the same column.
`y`	A `character` giving the name of the column in `object` that contains the labels of the LSD values that are to be plotted as the rows.
`alpha`	A `numeric` giving the significance level for the LSD.
`triangles`	A `character` indicating whether the plot should include the `lower`, `upper` or `both` triangle(s). Here it is only used to adjust gridlines for the omission of the diagonal.
`gridspacing`	A `numeric` specifying the number(s) of rows and columns that form groups in the grid of differences. This is most useful when two or more factors index the rows and columns. If a single, nonzero number, *k* say, is given then a grid line is placed after every *k*th row and column. If a vector of values is given then the number of grid lines is the length of the vector and the spacing between each is specified by the elements of the vector.
`title`	A `character` string giving the main title for the plot.
`axis.labels`	A `character` string giving the label to use for both the x- and y-axis.
`axis.text.size`	A `numeric` giving the size of the `labels` on the axes of the `heatmap`.
`colours`	A vector of colours to be passed to the `ggplot` function `scale\_colour\_gradientn`.

ggplotFuncs	A list, each element of which contains the results of evaluating a ggplot2 function. It is created by calling the list function with a ggplot2 function call for each element. These functions are applied in creating the ggplot object.
printPlot	A logical indicating whether or not the a plot is to be printed. This would be used when just the returned ggplot object is required.
...	Provision for passing arguments to functions called internally - not used at present.

Value

An object of class "ggplot", which can be plotted using print or otherwise manipulated.

Author(s)

Chris Brien

See Also

plotLSDs.alldiffs, plotLSDerrors.alldiffs, plotLSDerrors.data.frame, exploreLSDs, ggplot

Examples

```
##Subset WaterRunoff data to reduce time to execute
data(WaterRunoff.dat)
tmp <- subset(WaterRunoff.dat, Date == "05-18")

##Use asreml to get predictions and associated statistics

## Not run:
asreml.options(keep.order = TRUE) #required for asreml-R4 only
current.asr <- asreml(fixed = pH ~ Benches + (Sources * (Type + Species)),
                      random = ~ Benches:MainPlots,
                      keep.order=TRUE, data= tmp))
current.asrt <- as.asrtests(current.asr, NULL, NULL)
TS.diffs <- predictPlus.asreml(classify = "Sources:Type",
                               asreml.obj = current.asr, tables = "none",
                               wald.tab = current.asrt$wald.tab,
                               present = c("Type","Species","Sources"))

## End(Not run)

## Use lmeTest and emmmeans to get predictions and associated statistics

if (requireNamespace("lmerTest", quietly = TRUE) &
    requireNamespace("emmeans", quietly = TRUE))
{
  m1.lmer <- lmerTest::lmer(pH ~ Benches + (Sources * (Type + Species)) +
                              (1|Benches:MainPlots),
                            data=na.omit(WaterRunoff.dat))
  TS.emm <- emmeans::emmeans(m1.lmer, specs = ~ Sources:Type)
```

```
  TS.preds <- summary(TS.emm)
  den.df <- min(TS.preds$df, na.rm = TRUE)
  ## Modify TS.preds to be compatible with a predictions.frame
  TS.preds <- as.predictions.frame(TS.preds, predictions = "emmean",
                                   se = "SE", interval.type = "CI",
                                   interval.names = c("lower.CL", "upper.CL"))

  ## Form an all.diffs object and check its validity
  TS.vcov <- vcov(TS.emm)
  TS.diffs <- allDifferences(predictions = TS.preds, classify = "Sources:Type",
                             vcov = TS.vcov, tdf = den.df)
  validAlldiffs(TS.diffs)
}

## Plot LSD values for predictions obtained using asreml or lmerTest
if (exists("TS.diffs"))
{
  LSD <- within(reshape2::melt(TS.diffs$p.differences),
              {
                Var1 <- factor(Var1, levels=dimnames(TS.diffs$p.differences)[[1]])
                Var2 <- factor(Var2, levels=levels(Var1))
              })
  names(LSD) <- c("Rows","Columns","LSDs")
  plotLSDs(LSD, x = "Rows", y = "Columns", gridspacing = rep(c(3,4), c(4,2)))
}
```

`plotPredictions.data.frame` *Plots the predictions for a term, possibly with error bars.*

Description

This function plots the predictions `y` that are based on `classify` and stored in the `data.frame` `data`. The package `ggplot2` is used to produce the plots. Line plots are produced when variables involving `x.num` or `x.fac` are involved in `classify` for the predictions; otherwise, bar charts are produced. Further, for line charts, the argument `panels` determines whether a single plot or multiple plots in a single window are produced; for bar charts, the argument `panels` is ignored.

Usage

```
## S3 method for class 'data.frame'
plotPredictions(data, classify, y,
                x.num = NULL, x.fac = NULL, nonx.fac.order = NULL,
                colour.scheme = "colour", panels = "multiple",
                graphics.device = NULL,
                error.intervals = "Confidence", interval.annotate = TRUE,
                titles = NULL, y.title = NULL,
                filestem = NULL, printPlot = TRUE, ggplotFuncs = NULL, ...)
```

Arguments

data	A predictions.frame, or data.frame, containing the values of the variables to be plotted. Generally, it should contain the variables classifying the predictions and include a column with the name specified in the y argument, usually predicted.value or backtransformed.predictions; each row contains a single predicted value. It should also include columns for the standard.error and est.status. The number of rows should equal the number of unique combinations of the classifying variables. While such a data.frame can be constructed from the beginning, the pvals component of the value produced by predict.asreml is a suitable value to supply for this argument. Note that the names standard.error and est.status have been changed to std.error and status in the pvals component produced by asreml-R4; if the new names are in the data.frame supplied to predictions, they will be returned to the previous names. If error.intervals is not "none", then the predictions component and, if present, the backtransforms component should contain columns for the lower and upper values of the limits for the interval with names that begin with lower and upper, respectively. The second part of the name must be one of Confidence, StandardError or halfLeastSignificant. The last part needs to be consistent between the lower and upper limits.
classify	A character string giving the combinations of the independent variables on which the predictions are based. It is an interaction type term formed from the independent variables, that is, separating the variable names with the : operator. To predict the overall mean, set the classify to "(Intercept)".
y	A character string giving the name of the variable that is to be plotted on the Y axis.
x.num	A character string giving the name of the numeric covariate that (i) is potentially included in terms in the fitted model and (ii) is the x-axis variable for plots. Its values will not be converted to a factor.
x.fac	A character string giving the name of the factor that corresponds to x.num, is potentially included in terms in the fitted model and which corresponds to the x-axis variable. It should have the same number of levels as the number of unique values in x.num. The levels of x.fac must be in the order in which they are to be plotted - if they are dates, then they should be in the form yyyymmdd, which can be achieved using as.Date. However, the levels can be non-numeric in nature, provided that x.num is also set.
nonx.fac.order	A character vector giving the order in which factors other than x.fac are to be plotted in faceted plots (i.e. where the number of non x factors is greater than 1). The first factor in the vector will be plotted on the X axis (if there is no x.num or x.fac. Otherwise, the order of plotting the factors is in columns (X facets) and then rows (Y facets). By default the order is in decreasing order for the numbers of levels of the non x factors.
colour.scheme	A character string specifying the colour scheme for the plots. The default is "colour" which produces coloured lines and bars, a grey background and white gridlines. A value of "black" results in black lines, grey bars and gridlines and a white background.

`panels` Possible values are `"single"` and `"multiple"`. When line plots are to be produced, because variables involving `x.num` or `x.fac` are involved in `classify` for the predictions, `panels` determines whether or not a single panel or multiple panels in a single window are produced. The `panels` argument is ignored for bar charts.

`graphics.device`
A `character` specifying a graphics device for plotting. The default is `graphics.device = NULL`, which will result in plots being produced on the current graphics device. Setting it to `"windows"`, for example, will result in a windows graphics device being opened.

`error.intervals`
A character string indicating the type of error interval, if any, to plot in order to indicate uncertainty in the results. Possible values are `"none"`, `"StandardError"`, `"Confidence"` and `"halfLeastSignificant"`. Here, any option other than `"none"` will result in the interval limits contained in `data` being plotted.

`interval.annotate`
A `logical` indicating whether the plot annotation indicating the type of `error.interval` is to be included in the plot.

`titles` A `list`, each component of which is named for a column in the `data.frame` for the `asreml.obj` used in making the predictions and contains a `character string` giving a title to use in output (e.g. tables and graphs). Here they will be used for axis labels for nonresponse variables. For response variable labels see `y.title`.

`y.title` The title to be displayed on the y axis of any plot.

`filestem` A character sting giving the beginning of the name of the file in which to save the plot. If `filestem = NULL`, the plot is not saved. The remainder of the file name will be generated automatically and consists of the following elements separated by full stops: the classify term, `Bar` or `Line` and, if `error.intervals` is not `"none"`, one of `SE`, `CI` or `LSI`. The file will be saved as a 'png' file in the current work directory.

`printPlot` A `logical` indicating whether or not the a plot is to be printed. This would be used when just the returned `ggplot` object is required.

`ggplotFuncs` A `list`, each element of which contains the results of evaluating a `ggplot2` function. It is created by calling the `list` function with a `ggplot2` function call for each element. These functions are applied in creating the `ggplot` object for plotting.

`...` further arguments passed to `ggplot`.

Value

An object of class `ggplot`, which can be plotted using `print` or otherwise manipulated.

Author(s)

Chris Brien

See Also

allDifferences.data.frame, predictPresent.asreml, redoErrorIntervals.alldiffs, recalcLSD.alldiffs, ggplot, Devices

Examples

```
## Not run:
data(WaterRunoff.dat)
asreml.options(keep.order = TRUE) #required for asreml-R4 only
current.asr <- asreml(fixed = log.Turbidity ~ Benches + Sources + Type + Species +
                                Sources:Type + Sources:Species +
                                Sources:xDay + Species:xDay + Species:Date,
                      data = WaterRunoff.dat, keep.order = TRUE)
current.asrt <- as.asrtests(current.asr, NULL, NULL)

#### Get the observed combinations of the factors and variables in classify
class.facs <- c("Species","Date","xDay")
levs <- as.data.frame(table(WaterRunoff.dat[class.facs]))
levs <- as.list(levs[levs$Freq != 0, class.facs])
levs$xDay <- as.numfac(levs$xDay)

predictions <- predict(current.asr, classify="Species:Date:xDay",
                       parallel = TRUE, levels = levs,
                       present = c("Type","Species","Sources"))
#### for asreml-R3
predictions <- predictions$predictions$pvals
predictions <- predictions[predictions$est.status == "Estimable",]
#### for asreml-R4
predictions <- predictions$pvals
predictions <- predictions[predictions$status == "Estimable",]
#### end
plotPredictions(classify="Species:Date:xDay", y = "predicted.value",
                data = predictions,
                x.num = "xDay", x.fac = "Date",
                x.title = "Days since first observation",
                y.title = "Predicted log(Turbidity)",
                present = c("Type","Species","Sources"),
                error.intervals = "none",
                ggplotFuncs = list(ggtitle("Transformed turbidity over time")))

diffs <- predictPlus(classify="Species:Date:xDay",
                     present=c("Type","Species","Sources"),
                     asreml.obj = current.asr, tables = "none",
                     x.num = "xDay", x.fac = "Date",
                     parallel = TRUE, levels = levs,
                     x.plot.values=c(0,28,56,84),
                     wald.tab = current.asrt$wald.tab)
x.title <- "Days since first observation"
names(x.title) <- "xDay"
plotPredictions(classify="Species:Date:xDay", y = "predicted.value",
                data = diffs$predictions,
                x.num = "xDay", x.fac = "Date",
```

```
                titles = x.title,
                y.title = "Predicted log(Turbidity)")

## End(Not run)

## Use lmerTest and emmmeans to get predictions and associated statistics
if (requireNamespace("lmerTest", quietly = TRUE) &
    requireNamespace("emmeans", quietly = TRUE))
{
  data(Ladybird.dat)
  m1.lmer <- lmerTest::lmer(logitP ~ Host*Cadavers*Ladybird + (1|Run),
                            data=Ladybird.dat)
  HCL.emm <- emmeans::emmeans(m1.lmer, specs = ~ Host:Cadavers:Ladybird)
  HCL.preds <- summary(HCL.emm)
  den.df <- min(HCL.preds$df)
  ## Modify HCL.preds to be compatible with a predictions.frame
  HCL.preds <- as.predictions.frame(HCL.preds, predictions = "emmean",
                                    se = "SE", interval.type = "CI",
                                    interval.names = c("lower.CL", "upper.CL"))
  ## Plot the predictions
  plotPredictions(HCL.preds, y = "predicted.value", "Host:Cadavers:Ladybird")
}
```

`plotPvalues.alldiffs` *Plots a heat map of p-values for pairwise differences between predictions.*

Description

Produces a heat-map plot of the p-values for pairwise differences between predictions that is stored in the `p.differences` component of an `all.diffs` object. This is generally a matrix whose rows and columns are labelled by the levels of one or more factors, the set of labels being the same for rows and columns. The `sections` argument allows multiple plots to be produced, one for each combination of the levels of the factors listed in `sections`. Otherwise, a single plot is produced for all observed combinations of the levels of the factors in the `classify` for the `alldiffs.object`. The plots are produced using `plotPvalues.data.frame`. The order of plotting the levels of one of the factors indexing the predictions can be modified using `sort.alldiffs`.

Usage

```
plotPvalues(object, ...)
## S3 method for class 'alldiffs'
plotPvalues(object, sections = NULL,
            gridspacing = 0, factors.per.grid = 0,
            show.sig = FALSE, alpha = 0.10,
            sig.size = 3, sig.colour = "black",
            sig.face = "plain", sig.family = "",
            triangles = "both",
            title = NULL, axis.labels = TRUE, axis.text.size = 12,
```

```
sep=",", colours = RColorBrewer::brewer.pal(3, "Set2"),
ggplotFuncs = NULL, printPlot = TRUE,
sortFactor = NULL, sortParallelToCombo = NULL,
sortNestingFactor = NULL, sortOrder = NULL,
decreasing = FALSE, ...)
```

Arguments

object	An alldiffs.object with a p.differences component that is not NULL.
sections	A character listing the names of the factors that are to be used to break the plot into sections. A separate plot will be produced for each observed combination of the levels of these factors.
gridspacing	A numeric specifying the number(s) of rows and columns that form groups in the grid of differences. An alternative is to specify the factors.per.grid argument to have the grid spacings automatically calculated. Grids are most useful when two or more factors index the rows and columns. If a single, nonzero number, *k* say, is given then a grid line is placed after every *k*th row and column. If a vector of values is given then the number of grid lines is the length of the vector and the spacing between each is specified by the elements of the vector.
factors.per.grid	A numeric specifying the number of factors to include within each grid of differences. The gridspacing will then be computed based on the numbers of combinations observed within the levels of the remaining factors in a single plot. The gridspacing argument to this function will be ignored if factors.per.grid is greater than zero. Grids are most useful when two or more factors index the rows and columns of each plot.
show.sig	A logical that specifies whether asterisks indicating the level of significance are to be added to the plot. If they are then '***' indicates that $p \leq 0.001$, '**' that $0.001 < p \leq 0.01$, '*' that $0.01 < p \leq 0.05$ '.' that $0.05 < p \leq 0.10$. The last is only included for alpha = 0.10.
alpha	A numeric giving the significance level for testing pairwise differences; must be 0.05 or 0.10.
sig.size	A numeric specifying the size, in pts, of the significance asterisks.
sig.colour	A character specifying the colour to use for the significance asterisks.
sig.face	A character specifying the font face for the significance asterisks ("plain", "italic", "bold", "bold.italic").
sig.family	A character specifying the font family for the significance asterisks. The font families that are available depends on the system. For font families other than the basic Postscript fonts, see the extrafont package.
triangles	A character indicating whether the plot should include the lower, upper or both traingle(s).
title	A character string giving the main title for the plot and to which is appended the levels combination of the sectioning factors, if any, for each plot.
axis.labels	A logical indicating whether a label is to be added to the x- and y-axes. If TRUE, the label is the comma-separated list of factors whose levels combinations are involved in the prediction differences for which the p-values are calculated.

axis.text.size A numeric giving the size of the labels on the axes of the heatmap.

sep A character giving the characters separating the levels of different factors in the row and column names of the p.differences component.

colours A vector of of colours to be passed to the ggplot function scale_colour_gradientn.

ggplotFuncs A list, each element of which contains the results of evaluating a ggplot2 function. It is created by calling the list function with a ggplot2 function call for each element. It is passed to ggplot via plotPvalues.data.frame to be applied in creating the ggplot object.

printPlot A logical indicating whether or not the a plot is to be printed. This would be used when just the returned data.frame is required.

sortFactor A character containing the name of the factor that indexes the set of predicted values that determines the sorting of the components. If there is only one variable in the classify term then sortFactor can be NULL and the order is defined by the complete set of predicted values. If there is more than one variable in the classify term then sortFactor must be set. In this case the sortFactor is sorted in the same order within each combination of the values of the sortParallelToCombo variables: the classify variables, excluding the sortFactor. There should be only one predicted value for each unique value of sortFactor within each set defined by a combination of the values of the classify variables, excluding the sortFactor factor. The order to use is determined by either sortParallelToCombo or sortOrder.

sortParallelToCombo

A list that specifies a combination of the values of the factors and numerics, excluding sortFactor, that are in classify. Each of the components of the supplied list is named for a classify variable and specifies a single value for it. The combination of this set of values will be used to define a subset of the predicted values whose order will define the order of sortFactor. Each of the other combinations of the values of the factors and numerics will be sorted in parallel. If sortParallelToCombo is NULL then the first value of each classify variable, except for the sortFactor factor, in the predictions component is used to define sortParallelToCombo. If there is only one variable in the classify then sortParallelToCombo is ignored.

sortNestingFactor

A character containing the name of the factor that defines groups of the sortFactor within which the predicted values are to be ordered. If there is only one variable in the classify then sortNestingFactor is ignored.

sortOrder A character vector whose length is the same as the number of levels for sortFactor in the predictions component of the alldiffs.object. It specifies the desired order of the levels in the reordered components of the alldiffs.object. The argument sortParallelToCombo is ignored.

The following creates a sortOrder vector levs for factor f based on the values in x: levs <- levels(f)[order(x)].

decreasing A logical passed to order that detemines whether the order for sorting the alldiffs.object components is for increasing or decreasing magnitude of the predicted values.

... Provision for passsing arguments to functions called internally - not used at present.

Value

A list with components named pvalues and plots. The pvalues component contains the data.frame with the columns Rows, Columns, p, sections1 and sections2. This data.frame is formed using the sed component of object and is used by plotPvalues.data.frame in producng the plot. The plots component contains a list of ggplot objects, one for each plot produced. Multiple plots are stored in the plots component if the sections argument is set and the plots are are named for the levels combinations of the sections.

Author(s)

Chris Brien

See Also

plotPvalues.data.frame, allDifferences.data.frame, sort.alldiffs, subset.alldiffs, ggplot

Examples

```
##Subset WaterRunoff data to reduce time to execute
data(WaterRunoff.dat)
tmp <- subset(WaterRunoff.dat, Date == "05-18" & Benches != "3")

##Use asreml to get predictions and associated statistics

## Not run:
asreml.options(keep.order = TRUE) #required for asreml-R4 only
current.asr <- asreml(fixed = pH ~ Benches + (Sources * (Type + Species)),
                      random = ~ Benches:MainPlots,
                      keep.order=TRUE, data= tmp)
current.asrt <- as.asrtests(current.asr, NULL, NULL)
TS.diffs <- predictPlus.asreml(classify = "Sources:Type",
                               asreml.obj = current.asr, tables = "none",
                               wald.tab = current.asrt$wald.tab,
                               present = c("Type","Species","Sources"))

## End(Not run)

## Use lmeTest and emmmeans to get predictions and associated statistics

if (requireNamespace("lmerTest", quietly = TRUE) &
    requireNamespace("emmeans", quietly = TRUE))
{
  m1.lmer <- lmerTest::lmer(pH ~ Benches + (Sources * (Type + Species)) +
                              (1|Benches:MainPlots),
                            data=na.omit(WaterRunoff.dat))
  TS.emm <- emmeans::emmeans(m1.lmer, specs = ~ Sources:Type)
  TS.preds <- summary(TS.emm)
  den.df <- min(TS.preds$df, na.rm = TRUE)
  ## Modify TS.preds to be compatible with a predictions.frame
  TS.preds <- as.predictions.frame(TS.preds, predictions = "emmean",
```

```
                                    se = "SE", interval.type = "CI",
                                    interval.names = c("lower.CL", "upper.CL"))

  ## Form an all.diffs object and check its validity
  TS.vcov <- vcov(TS.emm)
  TS.diffs <- allDifferences(predictions = TS.preds, classify = "Sources:Type",
                             vcov = TS.vcov, tdf = den.df)
  validAlldiffs(TS.diffs)
}

## Plot p-values for predictions obtained using asreml or lmerTest
if (exists("TS.diffs"))
{
  plotPvalues(TS.diffs, gridspacing = rep(c(3,4), c(4,2)), show.sig = TRUE)

  plotPvalues(TS.diffs, sections = "Sources", show.sig = TRUE, axis.labels = TRUE)

}
```

`plotPvalues.data.frame` *Plots a heat map of p-values for pairwise differences between predictions.*

Description

Produces a heat-map plot of the p-values for pairwise differences between predictions that is in a `data.frame`. The `data.frame` includes two factors whose levels specify, for each p-value, which factor levels are being compared.

Usage

```
## S3 method for class 'data.frame'
plotPvalues(object, p = "p", x, y,
            gridspacing = 0, show.sig = FALSE, alpha = 0.10,
            sig.size = 3, sig.colour = "black",
            sig.face = "plain", sig.family = "",
            triangles = "both",
            title = NULL, axis.labels = NULL, axis.text.size = 12,
            colours = RColorBrewer::brewer.pal(3, "Set2"),
            ggplotFuncs = NULL, printPlot = TRUE, ...)
```

Arguments

`object`	A `data.frame` containing the three columns specified by `p`, `x` and `y`.
`p`	A `character` giving the name of the column in `object` that contains the p-values to be plotted.

x	A character giving the name of the column in object that contains the factor whose levels index the p-values that are to be plotted in the same column.
y	A character giving the name of the column in object that contains the labels of the p-values that are to be plotted as the rows.
gridspacing	A numeric specifying the number(s) of rows and columns that form groups in the grid of differences. This is most useful when two or more factors index the rows and columns. If a single, nonzero number, *k* say, is given then a grid line is placed after every *k*th row and column. If a vector of values is given then the number of grid lines is the length of the vector and the spacing between each is specified by the elements of the vector.
show.sig	A logical that specifies whether asterisks indicating the level of significance are to be added to the plot. If they are then '***' indicates that $p \leq 0.001$, '**' that $0.001 < p \leq 0.01$, '*' that $0.01 < p \leq 0.05$ '.' that $0.05 < p \leq 0.10$. The last is only included for alpha = 0.10.
alpha	A numeric giving the significance level for testing pairwise differences; must be 0.05 or 0.10.
sig.size	A numeric specifying the size, in pts, of the significance asterisks.
sig.colour	A character specifying the colour to use for the significance asterisks.
sig.face	A character specifying the font face for the significance asterisks ("plain", "italic", "bold", "bold.italic").
sig.family	A character specifying the font family for the significance asterisks. The font families that are available depends on the system. For font families other than the basic Postscript fonts, see the extrafont package.
triangles	A character indicating whether the plot should include the lower, upper or both traingle(s). Here it is only used to adjust gridlines for the omission of the diagonal.
title	A character string giving the main title for the plot.
axis.labels	A character string giving the label to use for both the x- and y-axis.
axis.text.size	A numeric giving the size of the labels on the axes of the heatmap.
colours	A vector of of colours to be passed to the ggplot function scale_colour_gradientn.
ggplotFuncs	A list, each element of which contains the results of evaluating a ggplot2 function. It is created by calling the list function with a ggplot2 function call for each element. These functions are applied in creating the ggplot object.
printPlot	A logical indicating whether or not the a plot is to be printed. This would be used when just the returned ggplot object is required.
...	Provision for passsing arguments to functions called internally - not used at present.

Value

An object of class "ggplot", which can be plotted using print or otherwise manipulated.

Author(s)

Chris Brien

See Also

plotPvalues.alldiffs, allDifferences.data.frame, ggplot

Examples

```
##Subset WaterRunoff data to reduce time to execute
data(WaterRunoff.dat)
tmp <- subset(WaterRunoff.dat, Date == "05-18")

##Use asreml to get predictions and associated statistics

## Not run:
asreml.options(keep.order = TRUE) #required for asreml-R4 only
current.asr <- asreml(fixed = pH ~ Benches + (Sources * (Type + Species)),
                      random = ~ Benches:MainPlots,
                      keep.order=TRUE, data= tmp))
current.asrt <- as.asrtests(current.asr, NULL, NULL)
SS.diffs <- predictPlus.asreml(classify = "Sources:Type",
                               asreml.obj = current.asr, tables = "none",
                               wald.tab = current.asrt$wald.tab,
                               present = c("Type","Species","Sources"))

## End(Not run)

## Use lmeTest and emmmeans to get predictions and associated statistics

if (requireNamespace("lmerTest", quietly = TRUE) &
    requireNamespace("emmeans", quietly = TRUE))
{
  m1.lmer <- lmerTest::lmer(pH ~ Benches + (Sources * (Type + Species)) +
                              (1|Benches:MainPlots),
                            data=na.omit(WaterRunoff.dat))
  TS.emm <- emmeans::emmeans(m1.lmer, specs = ~ Sources:Type)
  TS.preds <- summary(TS.emm)
  den.df <- min(TS.preds$df, na.rm = TRUE)
  ## Modify TS.preds to be compatible with a predictions.frame
  TS.preds <- as.predictions.frame(TS.preds, predictions = "emmean",
                                   se = "SE", interval.type = "CI",
                                   interval.names = c("lower.CL", "upper.CL"))

  ## Form an all.diffs object and check its validity
  TS.vcov <- vcov(TS.emm)
  TS.diffs <- allDifferences(predictions = TS.preds, classify = "Sources:Type",
                             vcov = TS.vcov, tdf = den.df)
  validAlldiffs(TS.diffs)
}

## Plot p-values for predictions obtained using asreml or lmerTest
if (exists("TS.diffs"))
{
  p <- within(reshape2::melt(TS.diffs$p.differences),
              {
```

```
                Var1 <- factor(Var1, levels=dimnames(TS.diffs$p.differences)[[1]])
                Var2 <- factor(Var2, levels=levels(Var1))
              })
  names(p) <- c("Rows","Columns","p")
  plotPvalues(p, x = "Rows", y = "Columns",
              gridspacing = rep(c(3,4), c(4,2)), show.sig = TRUE)
}
```

`plotVariofaces.data.frame` *Plots empirical variogram faces, including envelopes, from supplied residuals as described by Stefanova, Smith & Cullis (2009).*

Description

Produces a plot for each face of an empirical 2D `variogram` based on supplied residuals from both an observed data set and simulated data sets. Those from simulated data sets are used to produce confidence envelopes If the data consists of sections, such as separate experiments, the two variogram faces are produced for each section. This function is less efficient in storage terms than `variofaces.asreml`, because here the residuals from all simulated data sets must be saved, in addition to the values for the variogram faces; in `variofaces.asreml`, the residuals for each simulated data set are discarded after the variogram has been calculated. On the other hand, the present function is more flexible, because there is no restriction on how the residuals are obtained.

Usage

```
## S3 method for class 'data.frame'
plotVariofaces(data, residuals, restype="Residuals", ...)
```

Arguments

`data`	A `data.frame` with either 3 or 4 columns. Only if there are 4 columns, the first should be a factor indexing sections for which separate variogram plots are to be produced. In either case, the other 3 columns should be, in order, (i) a factor indexing the x-direction, (ii) a factor indexing the y-direction, and (iii) the residuals for the observed response.
`residuals`	A `data.frame`, with either 2 or 3 initial columns followed by columns, each of which are the residuals from a simulated data set.
`restype`	A `character` describing the type of residuals that have been supplied. It will be used in the plot titles.
`...`	Other arguments that are passed down to the function `asreml.variogram`.

Details

For each set of residuals, `asreml.variogram` is used to obtain the empirical variogram, from which the values for its faces are obtained. Plots are produced for each face and include the observed residuals and the 2.5%, 50% & 97.5% quantiles.

Value

A `list` with the following components:

1. **face1:** a `data.frame` containing the variogram values on which the plot for the first dimension is based.
2. **face2:** a `data.frame` containing the variogram values on which the plot for the second dimension is based.

Author(s)

Chris Brien

References

Stefanova, K. T., Smith, A. B. & Cullis, B. R. (2009) Enhanced diagnostics for the spatial analysis of field trials. *Journal of Agricultural, Biological, and Environmental Statistics*, **14**, 392–410.

See Also

asremlPlus-package, asreml, asreml.variogram, variofaces.asreml, simulate.asreml.

Examples

```
## Not run:
data(Wheat.dat)
current.asr <- asreml(yield ~ Rep + WithinColPairs + Variety,
                      random = ~ Row + Column + units,
                      residual = ~ ar1(Row):ar1(Column),
                      data=Wheat.dat)
current.asrt <- as.asrtests(current.asr, NULL, NULL)
current.asrt <- rmboundary.asrtests(current.asrt)
# Form variance matrix based on estimated variance parameters
s2 <- current.asr$sigma2
gamma.Row <- current.asr$gammas[1]
gamma.unit <- current.asr$gammas[2]
rho.r <- current.asr$gammas[4]
rho.c <- current.asr$gammas[5]
row.ar1 <- mat.ar1(order=10, rho=rho.r)
col.ar1 <- mat.ar1(order=15, rho=rho.c)
V <- gamma.Row * fac.sumop(Wheat.dat$Row) +
  gamma.unit * diag(1, nrow=150, ncol=150) +
  mat.dirprod(col.ar1, row.ar1)
V <- s2*V

#Produce variogram faces plot (Stefanaova et al, 2009)
resid <- simulate(current.asr, V=V, which="residuals")
resid$residuals <- cbind(resid$observed[c("Row","Column")],
                         resid$residuals)
plotVariofaces(data=resid$observed[c("Row","Column","residuals")],
               residuals=resid$residuals,
               restype="Standardized conditional residuals")
```

```
## End(Not run)
```

`powerTransform` *Performs a combination of a linear and a power transformation on a variable. The transformed variable is stored in the* `data.frame` `data`.

Description

Perform a combination of a linear and a power transformation on a variable whose name is given as a character string in `var.name`. The transformed variable is stored in the `data.frame` `data`. The name of the transformed variable is made by prepending to the original `var.name` a combination of (i) `.offset`, if `offset` is nonzero, (ii) `neg.`, if `scale` is -1, or `scaled.`, if `abs(scale)` is other than one, and (iii) either `log.`, `sqrt.`, `recip.` or `power.`, if `power` is other than one. No action is taken if there is no transformation (i.e. `offset = 0`, `scale = 1` and `power = 1`). Also, the `titles` list is extended to include a component with a generated title for the transformed variable with text indicating the transformation prepended to the `title` for the `var.name` obtained from the `titles list`. For nonzero `offset`, 'Offset ' is prepended, For `scaled` not equal to one, the possible prepends are 'Negative of ' and 'Scaled '. The possible prepended texts for `power` not equal to one are 'Logarithm of', 'Square root of ', 'Reciprocal of ' and 'Power nnnn of ', where `nnn` is the power used.

Usage

```
powerTransform(var.name, power = 1, offset = 0, scale = 1, titles = NULL, data)
```

Arguments

`var.name`	A `character string` specifying the name of the variable in the `data.frame` `data` that is to be transformed.
`power`	A number specifying the power to be used in the transformation. If equal to 1, the default, no power transformation is applied. Otherwise, the variable is raised to the specified `power`, after scaling and applying any nonzero `offset`. If `power = 0`, the natural logarithm is used to transform the response; however, if the smallest value to be log-transformed is less than 1e-04, an error is generated. A log-transformation in this situation may be possible if a nonzero `offset` and/or a `scale` not equal to one is used.
`offset`	A number to be added to each value of the variable, after any scaling and before applying any power transformation.
`scale`	A number to multiply each value of the variable, before adding any offset and applying any power transformation.
`titles`	A `character vector`, each element of which is named for a variable in `data` and is a `character string` giving a title to use in output (e.g. tables and graphs) involving the variable. If titles are not supplied, the column name of the variable in `data` is used.
`data`	A `data.frame` containing the variable to be transformed and to which the transformed variable is to be appended.

Value

A list with a component named data that is the data.frame containing the transformed variable, a component named tvar.name that is a character string that is the name of the transformed variable in data, and a component named titles that extends the list supplied in the titles argument to include a generated title for the transformed title, the name of the new component being tvar.name.

Author(s)

Chris Brien

See Also

angular, angular.mod.

Examples

```
## set up a factor with labels
x.dat <- data.frame(y = c(14, 42, 120, 150))

## transform y to logarithms
trans <- powerTransform("y", power = 0, titles=list(y = "Length (cm)"), data = x.dat)
x.dat <- trans$data
tvar.name <- trans$tvar.name

## transform y to logarithms after multiplying by -1 and adding 1.
z.dat <- data.frame( y = c(-5.25, -4.29, -1.22, 0.05))
trans <- powerTransform("y", power = 0, scale = -1, offset = 1 ,
                        titles=list(y = "Potential"), data = z.dat)
z.dat <- trans$data
tvar.name <- trans$tvar.name
```

---

predictions.frame *Description of a predictions object*

Description

A data.frame of S3-class predictions.frame that stores the predictions for a fitted model.

as.predictions.frame is function that converts a data.frame to an object of this class.

is.predictions.frame is the membership function for this class; it tests that an object has class predictions.frame.

validPredictionsFrame can be used to test the validity of a predictions.frame.

Value

A data.frame that begins with the variables classifying the predictions, in the same order as in the classify, followed by a column of predictions that is named either predicted.value or backtransformed.predictions; it also contains columns named standard.error and est.status. The number of rows should equal the number of unique combinations of the classifying variables. While such a data.frame can be constructed from the beginning, the pvals component of the value produced by predict.asreml is a suitable value to supply for this argument. Note that the names standard.error and est.status have been changed to std.error and status in the pvals component produced by asreml-R4; if the new names are in the data.frame supplied to predictions, they will be returned to the previous names.

The data.frame may also include columns for the lower and upper values of error intervals, either standard error, confidence or half-LSD intervals. The names of these columns will consist of three parts separated by full stops: 1) the first part will be lower or upper; 2) the second part will be one of Confidence, StandardError or halfLeastSignificant; 3) the third component will be limits.

IF accuracy.threshold is set to a numeric value at the time the prediction.frame is formed, it will also include a column logical values named LSDwarning.

When halfLeastSignificant limits have been included in a predictions.frame, its attributes will include those that are not NULL of LSDtype, LSDby, LSDstatistic, LSDaccuracy and LSDvalues. LSDvalues are the LSD values used to calculate the halfLeastSignificant error.intervals and are an expanded version of the values stored in the assignedLSD column of the LSD.frame.

See predictPlus.asreml for more information.

Author(s)

Chris Brien

See Also

predictPlus.asreml, is.predictions.frame, as.predictions.frame, validPredictionsFrame

Examples

```
data(Oats.dat)

## Use asreml to get predictions and associated statistics

## Not run:
m1.asr <- asreml(Yield ~ Nitrogen*Variety,
                 random=~Blocks/Wplots,
                 data=Oats.dat)
current.asrt <- as.asrtests(m1.asr)
Var.pred <- asreml::predict.asreml(m1.asr, classify="Nitrogen:Variety",
                                   sed=TRUE)
if (getASRemlVersionLoaded(nchar = 1) == "3")
  Var.pred <- Var.pred$predictions
Var.preds <- as.predictions.frame(Var.pred$pvals, se = "std.error",
                                  est.status = "status")
```

```
## End(Not run)

  ## Use lmerTest and emmmeans to get predictions and associated statistics
  if (requireNamespace("lmerTest", quietly = TRUE) &
      requireNamespace("emmeans", quietly = TRUE))
  {
    m1.lmer <- lmerTest::lmer(Yield ~ Nitrogen*Variety + (1|Blocks/Wplots),
                              data=Oats.dat)
    Var.emm <- emmeans::emmeans(m1.lmer, specs = ~ Nitrogen:Variety)
    Var.preds <- summary(Var.emm)
    Var.preds <- as.predictions.frame(Var.preds, predictions = "emmean",
                                      se = "SE", interval.type = "CI",
                                      interval.names = c("lower.CL", "upper.CL"))
  }

  if (exists("Var.preds"))
  {
    ## Check the class and validity of the alldiffs object
    is.predictions.frame(Var.preds)
    validPredictionsFrame(Var.preds)
  }
```

`predictPlus.asreml` *Forms the predictions for a term, their pairwise differences and associated statistics. A factor having parallel values may occur in the model and a linear transformation of the predictions can be specified. It results in an object of class alldifffs.*

Description

This function forms the predictions for `term` using `classify` and the supplied `asreml` object and stores them in an `alldiffs.object`. If `x.num` is supplied, the predictions will be obtained for the values supplied in `x.pred.values` and, if supplied, `x.plot.values` will replace them in the `alldiffs.object` that is returned. If `x.fac`, but not `x.num`, is specified, predictions will involve it and, if supplied, `x.plot.values` will replace the levels of `x.fac` in the `alldiffs.object` that is returned. In order to get the correct predictions you may need to supply additional arguments to `predict.asreml` through `...` e.g. `present`, `parallel`, `levels`. Any aliased predictions will be removed, as will any standard error of pairwise differences involving them.

Also calculated are the approximate degrees of freedom of the standard errors of the predictions. If the denominator degrees of freedom for `term` are available in `wald.tab`, they are used. Otherwise the residual degrees of freedom or the maximum of the denominator degrees in `wald.tab`, excluding the Intercept, are used. Which is used depends on the setting of `dDF.fault`. These degrees of freedom are used for the t-distribution on which p-values and confidence intervals are based. It is stored as an attribute to the `alldiffs.object`. The degrees of freedom are also used in calculating the minimum, mean and maximum LSD for comparing pairs of predictions, which are also stored in the `alldiffs.object`.

If `pairwise` = `TRUE`, all pairwise differences between the `predictions`, their standard errors, p-values and LSD statistics are computed using `allDifferences.data.frame`. This adds them to

the alldiffs.object as additional list components named differences, sed, p.differences and LSD.

If a linear transformation of the predictions is specified then the values of this linear transformation are returned, instead of the original predictions, along with their standard errors and the pairwise differences and associated statistics.

If a transformation has been applied in the analysis (any one of transform.power is not one, scale is not one and offset is nonzero), the backtransforms of the transformed values and their lower and upper error intervals are added to a data.frame that is consistent with the predictions data.frame. If transform.power is other than one, the standard.error column of the data.frame is set to NA. This data.frame is added to the alldiffs.object as a list component called backtransforms.

The printing of the components produced is controlled by the tables argument. The order of plotting the levels of one of the factors indexing the predictions can be modified and is achieved using sort.alldiffs.

Usage

```
## S3 method for class 'asreml'
predictPlus(asreml.obj, classify, term = NULL,
            inestimable.rm = TRUE,
            linear.transformation = NULL, EGLS.linTransform = TRUE,
            error.intervals = "Confidence", alpha = 0.05,
            wald.tab = NULL, dDF.fault = "residual",  dDF.values = NULL,
            pairwise = TRUE, Vmatrix = FALSE,
            avsed.tolerance = 0.25, accuracy.threshold = NA,
            LSDtype = "overall", LSDsupplied = NULL, LSDby = NULL,
            LSDstatistic = "mean", LSDaccuracy = "maxAbsDeviation",
            x.num = NULL, x.fac = NULL,
            x.pred.values = NULL, x.plot.values = NULL,
            titles = NULL,  tables = "all" , level.length = NA,
            transform.power = 1, offset = 0, scale = 1,
            transform.function = "identity",
            sortFactor = NULL, sortParallelToCombo = NULL,
            sortNestingFactor = NULL, sortOrder = NULL,
            decreasing = FALSE, trace = FALSE, ...)
```

Arguments

asreml.obj	asreml object for a fitted model.
classify	A character string giving the variables that define the margins of the multiway table to be predicted. Multiway tables are specified by forming an interaction type term from the classifying variables, that is, separating the variable names with the : operator. To predict the overall mean, set the classify to "(Intercept)".
term	A character string giving the variables that define the term that was fitted using asreml and that corresponds to classify. It only needs to be specified when it is different to classify; it is stored as an attribute of the alldiffs.object.

It is likely to be needed when the fitted model includes terms that involve both a numeric covariate and a factor that parallel each other; the classify would include the covariate and the term would include the factor.

inestimable.rm A logical indicating whether rows for predictions that are not estimable are to be removed from the components of the alldiffs.object.

linear.transformation

A formula or a matrix. If a formula is given then it is taken to be a submodel of a model term corresponding to the classify. The projection matrix that transforms the predictions so that they conform to the submodel is obtained; the submodel does not have to involve variables in the classify, but the variables must be columns in the predictions component of alldiffs.obj and the space for the submodel must be a subspace of the space for the term specified by the classify. For example, for classify set to "A:B", the submodel ~ A + B will result in the predictions for the combinations of A and B being made additive for the factors A and B. The submodel space corresponding to A + B is a subspace of the space A:B. In this case both the submodel and the classify involve only the factors A and B. To fit an intercept-only submodel, specify linear.transformation to be the formula ~1.

If a matrix is provided then it will be used to apply the linear transformation to the predictions. It might be a contrast matrix or a matrix of weights for a factor used to obtain the weighted average over that factor. The number of rows in the matrix should equal the number of linear combinations of the predictions desired and the number of columns should equal the number of predictions.

In either case, as well as the values of the linear combinations, their standard errors, pairwise differences and associated statistics are returned.

EGLS.linTransform

A logical indicating whether or not the linear.transformation of the predictions stored in an alldiffs.object by fitting a submodel supplied in a formula is to take into account the variance of the predictions using a Estimated Generalized Least Squares (EGLS) approach. This is likely to be appropriate when the variance matrix of the predictions is not compound symmetric i.e. when not all the variances are equal or not all the covariances are equal. If the variance matrix is compund symmetric, then the setting of EGLS.linTransform will not affect the transformed predictions.

error.intervals

A character string indicating the type of error interval, if any, to calculate in order to indicate uncertainty in the results. Possible values are "none", "StandardError", "Confidence" and "halfLeastSignificant". The default is for confidence limits to be used. The "halfLeastSignificant" option results in half the Least Significant Difference (LSD) being added and subtracted to the predictions, the LSD being calculated using the square root of the mean of the variances of all or a subset of pairwise differences between the predictions. If the LSD is zero, as can happen when predictions are constrained to be equal, then the limits of the error intervals are set to NA. If LSDtype is set to overall, the avsed.tolerance is not NA and the range of the SEDs divided by the average of the SEDs exceeds avsed.tolerance then the error.intervals calculations and the plotting will revert to confidence intervals.

alpha — A numeric giving the significance level for LSDs or one minus the confidence level for confidence intervals. It is stored as an attribute to the alldiffs.object.

wald.tab — A data.frame containing the pseudo-anova table for the fixed terms produced by a call to wald.asreml. The main use of it here is in determining the degrees of freedom for calculating confidence or half-LSD error.intervals and p-values, the latter to be stored in the p.differences component of the alldiffs.object that is created.

dDF.fault — A character specifying the method to use to obtain substitute denominator degrees of freedom. when the numeric or algebraic methods produce faulty values, viz. NA, Inf or less than 0.01. Consistent with when no denDF are available, the default is "residual" and so the residual degrees of freedom from asreml.obj$nedf are used. If dDF.fault = "none", no substitute denominator degrees of freedom are employed; if dDF.fault = "residual"; if dDF.fault = "maximum", the maximum of those denDF that are available, excluding that for the Intercept, is used; if all denDF are faulty, asreml.obj$nedf is used. If dDF.fault = "supplied", a vector of values for the denominator degrees of freedom is to be supplied in dDF.values. Any other setting is ignored and a warning message produced. Generally, substituting these degrees of freedom is anticonservative in that it is likely that the degrees of freedom used will be too large.

dDF.values — A vector of values to be used when dDF.fault = "supplied". Its values will be used when denDF in a test for a fixed effect is NA. This vector must be the same length as the number of fixed terms, including (Intercept) whose value could be NA.

pairwise — A logical indicating whether all pairwise differences of the predictions and their standard errors and p-values are to be computed and stored. If tables is equal to "differences" or "all" or error.intervals is equal to "halfLeastSignificant", they will be stored irrespective of the value of pairwise.

Vmatrix — A logical indicating whether the variance matrix of the predictions will be stored as a component of the alldiffs.object that is returned. If linear.transformation is set, it will be stored irrespective of the value of Vmatrix.

avsed.tolerance — A numeric giving the value of the SED range, the range of the SEDs divided by the square root of the mean of the variances of all or a subset of the pairwise differences, that is considered reasonable in calculating error.intervals. To have it ignored, set it to NA. It should be a value between 0 and 1. The following rules apply:

1. If avsed.tolerance is NA then mean LSDs of the type specified by LSDtype are calculated and used in error.intervals and plots.
2. Irrespective of the setting of LSDtype, if avsed.tolerance is not exceeded then the mean LSDs are used in error.intervals and plots.
3. If LSDtype is set to overall, avsed.tolerance is not NA, and avsed.tolerance is exceeded then error.intervals and plotting revert to confidence intervals.
4. If LSDtype is set to factor.combinations and avsed.tolerance is not exceeded for any factor combination then the half LSDs are used in error.intervals

and plots; otherwise, error.intervals and plotting revert to confidence intervals.

5. If LSDtype is set to per.prediction and avsed.tolerance is not exceeded for any prediction then the half LSDs are used in error.intervals and plots; otherwise, error.intervals and plotting revert to confidence intervals.

accuracy.threshold

A numeric specifying the value of the LSD accuracy measure, which measure is specified by LSDaccuracy, as a threshold value in determining whether the hallfLeastSignificant error.interval for a predicted value is a reasonable approximation; this will be the case if the LSDs across all pairwise comparisons for which the interval's LSD was computed, as specified by LSDtype and LSDby, are similar enough to the interval's LSD, as measured by LSDaccuracy. If it is NA, it will be ignored. If it is not NA, a column of logicals named LSDwarning will be added to the predictions component of the alldiffs.object. The value of LSDwarning for a predicted.value will be TRUE if the value of the LSDaccuracy measure computed from the LSDs for differences between this predicted.value and the other predicted.values as compared to its assignedLSD exceeds the value of accuracy.threshold. Otherwise, the value of LSDwarning for a predicted.value will be FALSE.

LSDtype

A character string that can be overall, factor.combinations, per.prediction or supplied. It determines whether the values stored in a row of a LSD.frame are the values calculated (i) overall from the LSD values for all pairwise comparisons, unless there is only one, possibly repeated, prediction, when a notional LSD is calculated, (ii) the values calculated from the pairwise LSDs for the levels of each factor.combination, unless there is only one prediction for a level of the factor.combination, when a notional LSD is calculated, (iii) per.prediction, being based, for each prediction, on all pairwise differences involving that prediction, or (iv) as supplied values of the LSD, specified with the LSDsupplied argument; these supplied values are to be placed in the assignedLSD column of the LSD.frame stored in an alldiffs.object so that they can be used in LSD calculations.

See LSD.frame for further information on the values in a row of this data.frame and how they are calculated.

LSDsupplied

A data.frame or a named numeric containing a set of LSD values that correspond to the observed combinations of the values of the LSDby variables in the predictions.frame or a single LSD value that is an overall LSD. If a data.frame, it may have (i) a column for the LSDby variable and a column of LSD values or (ii) a single column of LSD values with rownames being the combinations of the observed values of the LSDby variables. Any name can be used for the column of LSD values; assignedLSD is sensible, but not obligatory. Otherwise, a numeric containing the LSD values, each of which is named for the observed combination of the values of the LSDby variables to which it corresponds. (Applying the function dae::fac.combine to the predictions component is one way of forming the required combinations for the (row) names.) The values supplied will be incorporated into assignedLSD column of the LSD.frame stored as the LSD component of the alldiffs.object.

LSDby — A character (vector) of variables names, being the names of the factors or numerics in the classify; for each combination of their levels and values, there will be or is a row in the LSD.frame stored in the LSD component of the alldiffs.object when LSDtype is factor.combinatons.

LSDstatistic — A character nominating one or more of minimum, q10, q25, mean, median, q75, q90 or maximum as the value(s) to be stored in the assignedLSD column in an LSD.frame; the values in the assignedLSD column are used in computing halfLeastSignificant error.intervals. Here q10, q25, q75 and q90 indicate the sample quantiles corresponding to probabilities of 0.1, 0.25, 0.75 and 0.9 for the group of LSDs from which a single LSD value is calculated. The function quantile is used to obtain them. The mean LSD is calculated as the square root of the mean of the squares of the LSDs for the group. The median is calculated using the median function. Multiple values are only produced for LSDtype set to factor.combination, in which case LSDby must not be NULL and the number of values must equal the number of observed combinations of the values of the variables specified by LSDby. If LSDstatistic is NULL, it is reset to mean.

LSDaccuracy — A character nominating one of maxAbsDeviation, maxDeviation, q90Deviation or RootMeanSqDeviation as the statistic to be calculated as a measure of the accuracy of assignedLSD. The option q90Deviation produces the sample quantile corresponding to a probability of 0.90. The deviations are the differences between the LSDs used in calculating the LSD statistics and each assigned LSD and the accuracy is expressed as a proportion of the assigned LSD value. The calculated values are stored in the column named accuracyLSD in an LSD.frame.

titles — A list, each component of which is named for a column in the data.frame for asreml.obj and contains a character string giving a title to use in output (e.g. tables and graphs). Here they will be used for table headings.

tables — A character vector containing a combination of none, predictions, vcov, backtransforms, differences, p.differences, sed, LSD and all. These nominate which components of the alldiffs.object to print.

x.num — A character string giving the name of the numeric covariate that (i) is potentially included in terms in the fitted model and (ii) is the x-axis variable for plots. Its values will not be converted to a factor.

x.fac — A character string giving the name of the factor that (i) corresponds to x.num and (ii) is potentially included in terms in the fitted model. It should have the same number of levels as the number of unique values in x.num. The levels of x.fac must be in the order in which they are to be plotted - if they are dates, then they should be in the form yyyymmdd, which can be achieved using as.Date. However, the levels can be non-numeric in nature, provided that x.num is also set.

x.pred.values — The values of x.num for which predicted values are required. If levels is set for passing to predict.asreml, x.pred.values is ignored. Note that while levels is an alternative to x.pred.values, x.pred.values allows more general setting of the levels to be predicted.

x.plot.values — The actual values to be plotted on the x axis. They are needed when values different to those in x.num are to be plotted or x.fac is to be plotted because there is no x.num term corresponding to the same term with x.fac.

level.length The maximum number of characters from the levels of factors to use in the row and column labels of the tables of pairwise differences and their p-values and standard errors.

transform.power
A numeric specifying the power of a transformation, if one has been applied to the response variable. Unless it is equal to 1, the default, back-transforms of the predictions will be obtained and stored in the backtransforms component of the alldiffs.object. The back-transformation raises the predictions to the power equal to the reciprocal of transform.power, unless it equals 0 in which case the exponential of the predictions is taken.

offset A numeric that has been added to each value of the response after any scaling and before applying any power transformation.

scale A numeric by which each value of the response has been multiplied before adding any offset and applying any power transformation.

transform.function
A character giving the name of a function that specifies the scale on which the predicted values are defined. This may be the result of a transformation of the data using the function or the use of the function as a link function in the fitting of a generalized linear (mixed) model (GL(M)M). The possible transform.functions are identity, log, inverse, sqrt, logit, probit, and cloglog. The predicted.values and error.intervals, if not StandardError intervals, will be back-transformed using the inverse function of the transform.function. The standard.error column will be set to NA, unless (i) asreml returns columns named transformed.value and approx.se, as well as those called predicted.values and standard.error (such as when a GLM is fitted) and (ii) the values in transformed.value are equal to those obtained by backtransforming the predicted.values using the inverse function of the transform.function. Then, the approx.se values will be saved in the standard.error column of the backtransforms component of the returned alldiffs.obj. Also, the transformed.value and approx.se columns are removed from both the predictions and backtransforms components of the alldiffs.obj. Note that the values that end up in the standard errors column are approximate for the backtransformed values and are not used in calculating error.intervals.

sortFactor A character containing the name of the factor that indexes the set of predicted values that determines the sorting of the components. If there is only one variable in the classify term then sortFactor can be NULL and the order is defined by the complete set of predicted values. If there is more than one variable in the classify term then sortFactor must be set. In this case the sortFactor is sorted in the same order within each combination of the values of the sortParallelToCombo variables: the classify variables, excluding the sortFactor. There should be only one predicted value for each unique value of sortFactor within each set defined by a combination of the values of the classify variables, excluding the sortFactor factor. The order to use is determined by either sortParallelToCombo or sortOrder.

sortParallelToCombo
A list that specifies a combination of the values of the factors and numerics, excluding sortFactor, that are in classify. Each of the components of the

supplied list is named for a classify variable and specifies a single value for it. The combination of this set of values will be used to define a subset of the predicted values whose order will define the order of sortFactor. Each of the other combinations of the values of the factors and numerics will be sorted in parallel. If sortParallelToCombo is NULL then the first value of each classify variable, except for the sortFactor factor, in the predictions component is used to define sortParallelToCombo. If there is only one variable in the classify then sortParallelToCombo is ignored.

sortNestingFactor
: A character containing the name of the factor that defines groups of the sortFactor within which the predicted values are to be ordered. If there is only one variable in the classify then sortNestingFactor is ignored.

sortOrder
: A character vector whose length is the same as the number of levels for sortFactor in the predictions component of the alldiffs.object. It specifies the desired order of the levels in the reordered components of the alldiffs.object. The argument sortParallelToCombo is ignored.

  The following creates a sortOrder vector levs for factor f based on the values in x: levs <- levels(f)[order(x)].

decreasing
: A logical passed to order that detemines whether the order for sorting the components of the alldiffs.object is for increasing or decreasing magnitude of the predicted values.

trace
: A logical that control output from ASReml-R. If TRUE then partial iteration details are displayed when ASReml-R functions are invoked; if FALSE then no output is displayed.

...
: further arguments passed to predict.asreml.

Value

For linear.transformations set to NULL, an S3-class alldiffs.object with predictions and their standard errors and, depending on the settings of the arguments, all pairwise differences between predictions, their standard errors and p-values and LSD statistics. Also, unless the sortFactor or sortOrder arguments are invoked, the rows of predictions component are ordered so that they are in standard order for the variables in the classify. That is, the values of the last variable change with every row, those of the second-last variable only change after all the values of the last variable have been traversed; in general, the values of a variable are the same for all the combinations of the values to the variables to its right in the classify. In addition, if necessary, the order of the columns of the variables in the predictions component are changed to match their order in the classify.

If transform.power or scale is not one or offset is not zero, it will contain a data.frame with the backtransformed linear transformation of the predictions. The backtransformation will, after backtransforming for any power transformation, subtract the offset and then divide by the scale.

If error.intervals is not "none", then the predictions component and, if present, the backtransforms component will contain columns for the lower and upper values of the limits for the interval.

The name of the response, the response.title, the term, the classify, tdf, sortFactor and the sortOrder will be set as attributes to the object. Also, if error.intervals is "halfLeastSignificant", then those of LSDtype, LSDby and LSDstatistic that are not NULL will be added as attributes of

the object and of the predictions frame; additionally, LSDvalues will be added as attribute of the predictions frame, LSDvalues being the LSD values used in calculating the error.intervals. Note that the classify in an alldiffs.object is based on the variables indexing the predictions, which may differ from the classify used to obtain the original predictions (for example, when the alldiffs.objects stores a linear transformation of predictions.

For linear.transformations set to other than NULL, an alldiffs.object with the linear.transformation applied to the predictions and their standard errors and, depending on the settings of the arguments, all pairwise differences between the linearly transformed predictions, their standard errors and p-values and LSD statistics. (See also linTransform.alldiffs.)

Author(s)

Chris Brien

See Also

alldiffs.object, as.alldiffs, print.alldiffs, linTransform.alldiffs, sort.alldiffs, subset.alldiffs, allDifferences.data.frame, redoErrorIntervals.alldiffs, recalcLSD.alldiffs, exploreLSDs.alldiffs, pickLSDstatistics.alldiffs, predictPresent.asreml, plotPredictions.data.frame, as.Date, predict.asreml

Examples

```
## Not run:
data(WaterRunoff.dat)
asreml.options(keep.order = TRUE) #required for asreml-R4 only
current.asr <- asreml(fixed = pH ~ Benches + (Sources * (Type + Species)),
                      random = ~ Benches:MainPlots,
                      keep.order=TRUE, data= WaterRunoff.dat)
current.asrt <- as.asrtests(current.asr, NULL, NULL)
diffs <- predictPlus(classify = "Sources:Type",
                     asreml.obj = current.asr,
                     wald.tab = current.asrt$wald.tab,
                     present = c("Sources", "Type", "Species"))

## End(Not run)
```

---

predictPresent.asreml *Forms the predictions for each of one or more terms and presents them in tables and/or graphs.*

Description

This function forms the predictions for each term in terms using a supplied asreml object and predictPlus.asreml. Tables are produced using predictPlus.asreml, in conjunction with allDifferences.data.frame, with the argument tables specifying which tables are printed. The argument plots, along with transform.power, controls which plots are produced. The plots are produced using plotPredictions.data.frame, with line plots produced when variables involving

x.num or x.fac are involved in classify for the predictions and bar charts otherwise. In order to get the correct predictions you may need to supply additional arguments to predict.asreml through ... e.g. present, parallel, levels.

The order of plotting the levels of one of the factors indexing the predictions can be modified and is achieved using sort.alldiffs.

Usage

```
## S3 method for class 'asreml'
predictPresent(asreml.obj, terms, inestimable.rm = TRUE,
               linear.transformation = NULL, EGLS.linTransform,
               error.intervals = "Confidence", alpha = 0.05,
               wald.tab = NULL, dDF.fault = "residual", dDF.values = NULL,
               pairwise = TRUE, Vmatrix = FALSE,
               avsed.tolerance = 0.25, accuracy.threshold = NA,
               LSDtype = "overall", LSDsupplied = NULL, LSDby = NULL,
               LSDstatistic = "mean", LSDaccuracy = "maxAbsDeviation",
               x.num = NULL, x.fac = NULL, nonx.fac.order = NULL,
               x.pred.values = NULL, x.plot.values = NULL,
               plots = "predictions", panels = "multiple",
               graphics.device = NULL, interval.annotate = TRUE,
               titles = NULL, colour.scheme = "colour", save.plots = FALSE,
               transform.power = 1, offset = 0, scale = 1,
               transform.function = "identity",
               tables = "all", level.length = NA,
               sortFactor = NULL, sortParallelToCombo = NULL,
               sortNestingFactor = NULL, sortOrder = NULL,
               decreasing = FALSE,
               trace = FALSE, ggplotFuncs = NULL, ...)
```

Arguments

asreml.obj	asreml object for a fitted model.
terms	A character vector giving the terms for which predictions are required.
inestimable.rm	A logical indicating whether rows for predictions that are not estimable are to be removed from the components of the alldiffs.object.
linear.transformation	A formula or a matrix. If a formula is given then it is taken to be a submodel of a model term corresponding to the classify. The projection matrix that transforms the predictions so that they conform to the submodel is obtained; the submodel does not have to involve variables in the classify, but the variables must be columns in the predictions component of alldiffs.obj and the space for the submodel must be a subspace of the space for the term specified by the classify. For example, for classify set to "A:B", the submodel ~ A + B will result in the predictions for the combinations of A and B being made additive for the factors A and B. The submodel space corresponding to A + B is a subspace of the space A:B. In this case both the submodel and the clas-

sify involve only the factors A and B. To fit an intercept-only submodel, specify `linear.transformation` to be the formula `~1`.

If a `matrix` is provided then it will be used to apply the linear transformation to the `predictions`. It might be a contrast `matrix` or a `matrix` of weights for a factor used to obtain the weighted average over that factor. The number of rows in the `matrix` should equal the number of linear combinations of the `predictions` desired and the number of columns should equal the number of `predictions`.

In either case, as well as the values of the linear combinations, their standard errors, pairwise differences and associated statistics are returned in the `alldiffs.object`.

`EGLS.linTransform`

A `logical` indicating whether or not the `linear.transformation` of the predictions stored in an `alldiffs.object` by fitting a submodel supplied in a `formula` is to take into account the variance of the predictions using a Estimated Generalized Least Squares (EGLS) approach. This is likely to be appropriate when the variance matrix of the predictions is not compound symmetric i.e. when not all the variances are equal or not all the covariances are equal. If the variance matrix is compund symmetric, then the setting of `EGLS.linTransform` will not affect the transformed predictions.

`error.intervals`

A `character` string indicating the type of error interval, if any, to calculate in order to indicate uncertainty in the results. Possible values are `"none"`, `"StandardError"`, `"Confidence"` and `"halfLeastSignificant"`. The default is for confidence limits to be used. The `"halfLeastSignificant"` option results in half the Least Significant Difference (LSD) being added and subtracted to the predictions, the LSD being calculated using the square root of the mean of the variances of all or a subset of pairwise differences between the predictions. If the LSD is zero, as can happen when predictions are constrained to be equal, then the limits of the error intervals are set to `NA`. If `LSDtype` is set to `overall`, the `avsed.tolerance` is not `NA` and the range of the SEDs divided by the average of the SEDs exceeds `avsed.tolerance` then the `error.intervals` calculations and the plotting will revert to confidence intervals.

`alpha` A `numeric` giving the significance level for LSDs or one minus the confidence level for confidence intervals. It is stored as an attribute to the `alldiffs.object`.

`wald.tab` A `data.frame` containing the pseudo-anova table for the fixed terms produced by a call to `wald.asreml`. The main use of it here is in determining the degrees of freedom for calculating confidence or half-LSD `error.intervals` and p-values, the latter to be stored in the `p.differences` component of the `alldiffs.object` that is created.

`dDF.fault` A `character` specifying the method to use to obtain substitute denominator degrees of freedom. when the numeric or algebraic methods produce faulty values, viz. `NA`, `Inf` or less than 0.01. Consistent with when no denDF are available, the default is `"residual"` and so the residual degrees of freedom from `asreml.obj$nedf` are used. If `dDF.fault = "none"`, no substitute denominator degrees of freedom are employed; if `dDF.fault = "residual"`; if `dDF.fault = "maximum"`, the maximum of those denDF that are available, excluding that for the Intercept, is used; if all denDF are faulty, `asreml.obj$nedf` is used. If

dDF.fault = "supplied", a vector of values for the denominator degrees of freedom is to be supplied in dDF.values. Any other setting is ignored and a warning message produced. Generally, substituting these degrees of freedom is anticonservative in that it is likely that the degrees of freedom used will be too large.

dDF.values
: A vector of values to be used when dDF.fault = "supplied". Its values will be used when denDF in a test for a fixed effect is NA. This vector must be the same length as the number of fixed terms, including (Intercept) whose value could be NA.

pairwise
: A logical indicating whether all pairwise differences of the predictions and their standard errors and p-values are to be computed and stored. If tables is equal to "differences" or "all" or error.intervals is equal to "halfLeastSignificant", they will be stored irrespective of the value of pairwise.

Vmatrix
: A logical indicating whether the variance matrix of the predictions will be stored as a component of the alldiffs.object that is returned. If linear.transformation is set, it will be stored irrespective of the value of Vmatrix.

avsed.tolerance
: A numeric giving the value of the SED range, the range of the SEDs divided by the square root of the mean of the variances of all or a subset of the pairwise differences, that is considered reasonable in calculating error.intervals. It should be a value between 0 and 1. The following rules apply:

1. If avsed.tolerance is NA then mean LSDs of the type specified by LSDtype are calculated and used in error.intervals and plots.
2. Irrespective of the setting of LSDtype, if avsed.tolerance is not exceeded then the mean LSDs are used in error.intervals and plots.
3. If LSDtype is set to overall, avsed.tolerance is not NA, and avsed.tolerance is exceeded then error.intervals and plotting revert to confidence intervals.
4. If LSDtype is set to factor.combinations and avsed.tolerance is not exceeded for any factor combination then the half LSDs are used in error.intervals and plots; otherwise, error.intervals and plotting revert to confidence intervals.
5. If LSDtype is set to per.prediction and avsed.tolerance is not exceeded for any prediction then the half LSDs are used in error.intervals and plots; otherwise, error.intervals and plotting revert to confidence intervals.

accuracy.threshold
: A numeric specifying the value of the LSD accuracy measure, which measure is specified by LSDaccuracy, as a threshold value in determining whether the hallfLeastSignificant error.interval for a predicted value is a reasonable approximation; this will be the case if the LSDs across all pairwise comparisons for which the interval's LSD was computed, as specified by LSDtype and LSDby, are similar enough to the interval's LSD, as measured by LSDaccuracy. If it is NA, it will be ignored. If it is not NA, a column of logicals named LSDwarning will be added to the predictions component of the alldiffs.object. The value of LSDwarning for a predicted.value will be TRUE if the value of the LSDaccuracy measure computed from the LSDs for differences between

this predicted.value and the other predicted.values as compared to its assignedLSD exceeds the value of accuracy.threshold. Otherwise, the value of LSDwarning for a predicted.value will be FALSE.

LSDtype
: A character string that can be overall, factor.combinations, per.prediction or supplied. It determines whether the values stored in a row of a LSD.frame are the values calculated (i) overall from the LSD values for all pairwise comparisons, unless there is only one, possibly repeated, prediction, when a notional LSD is calculated, (ii) the values calculated from the pairwise LSDs for the levels of each factor.combination, unless there is only one prediction for a level of the factor.combination, when a notional LSD is calculated, (iii) per.prediction, being based, for each prediction, on all pairwise differences involving that prediction, or (iv) as supplied values of the LSD, specified with the LSDsupplied argument; these supplied values are to be placed in the assignedLSD column of the LSD.frame stored in an alldiffs.object so that they can be used in LSD calculations.

  See LSD.frame for further information on the values in a row of this data.frame and how they are calculated.

LSDsupplied
: A data.frame or a named numeric containing a set of LSD values that correspond to the observed combinations of the values of the LSDby variables in the predictions.frame or a single LSD value that is an overall LSD. If a data.frame, it may have (i) a column for the LSDby variable and a column of LSD values or (ii) a single column of LSD values with rownames being the combinations of the observed values of the LSDby variables. Any name can be used for the column of LSD values; assignedLSD is sensible, but not obligatory. Otherwise, a numeric containing the LSD values, each of which is named for the observed combination of the values of the LSDby variables to which it corresponds. (Applying the function dae::fac.combine to the predictions component is one way of forming the required combinations for the (row) names.) The values supplied will be incorporated into assignedLSD column of the LSD.frame stored as the LSD component of the alldiffs.object.

LSDby
: A character (vector) of variables names, being the names of the factors or numerics in the classify; for each combination of their levels and values, there will be or is a row in the LSD.frame stored in the LSD component of the alldiffs.object when LSDtype is factor.combinatons.

LSDstatistic
: A character nominating one or more of minimum, q10, q25, mean, median, q75, q90 or maximum as the value(s) to be stored in the assignedLSD column in an LSD.frame; the values in the assignedLSD column are used in computing halfLeastSignificant error.intervals. Here q10, q25, q75 and q90 indicate the sample quantiles corresponding to probabilities of 0.1, 0.25, 0.75 and 0.9 for the group of LSDs from which a single LSD value is calculated. The function quantile is used to obtain them. The mean LSD is calculated as the square root of the mean of the squares of the LSDs for the group. The median is calculated using the median function. Multiple values are only produced for LSDtype set to factor.combination, in which case LSDby must not be NULL and the number of values must equal the number of observed combinations of the values of the variables specified by LSDby. If LSDstatistic is NULL, it is reset to mean.

LSDaccuracy
: A character nominating one of maxAbsDeviation, maxDeviation, q90Deviation or RootMeanSqDeviation as the statistic to be calculated as a measure of the accuracy of assignedLSD. The option q90Deviation produces the sample quantile corresponding to a probability of 0.90. The deviations are the differences between the LSDs used in calculating the LSD statistics and each assigned LSD and the accuracy is expressed as a proportion of the assigned LSD value. The calculated values are stored in the column named accuracyLSD in an LSD.frame.

x.num
: A character string giving the name of the numeric covariate that (i) is potentially included in terms in the fitted model and (ii) is the x-axis variable for plots. Its values will not be converted to a factor.

x.fac
: A character string giving the name of the factor that (i) corresponds to x.num and (ii) is potentially included in terms in the fitted model. It should have the same number of levels as the number of unique values in x.num. The levels of x.fac must be in the order in which they are to be plotted - if they are dates, then they should be in the form yyyymmdd, which can be achieved using as.Date. However, the levels can be non-numeric in nature, provided that x.num is also set.

nonx.fac.order
: A character vector giving the order in which factors other than x.fac are to be plotted in plots with multiple panels (i.e. where the number of non-x factors is greater than 1). The first factor in the vector will be plotted on the X axis (if there is no x.num or x.fac. Otherwise, the order of plotting the factors is in columns (X facets) and then rows (Y facets). By default the order is in decreasing order for the numbers of levels of the non x factors.

x.pred.values
: The values of x.num for which predicted values are required.

x.plot.values
: The actual values to be plotted on the x axis or in the labels of tables. They are needed when values different to those in x.num are to be plotted or x.fac is to be plotted because there is no x.num term corresponding to the same term with x.fac.

plots
: Possible values are "none", "predictions", "backtransforms" and "both". Plots are not produced if the value is "none". If data are not transformed for analysis (transform.power = 1), a plot of the predictions is produced provided plots is not "none". If the data are transformed, the value of plots determines what is produced.

panels
: Possible values are "single" and "multiple". When line plots are to be produced, because variables involving x.num or x.fac are involved in classify for the predictions, panels determines whether or not a single panel or multiple panels in a single window are produced. The panels argument is ignored for bar charts.

graphics.device
: A character specifying a graphics device for plotting. The default is graphics.device = NULL, which will result in plots being produced on the current graphics device. Setting it to "windows", for example, will result in a windows graphics device being opened.

interval.annotate
: A logical indicating whether the plot annotation indicating the type of error.interval is to be included in the plot.

titles — A list, each component of which is named for a column in the data.frame for asreml.obj and contains a character string giving a title to use in output (e.g. tables and graphs). Here they will be used for axis labels.

colour.scheme — A character string specifying the colour scheme for the plots. The default is "colour" which produces coloured lines and bars, a grey background and white gridlines. A value of "black" results in black lines, grey bars and gridlines and a white background.

save.plots — A logical that determines whether any plots will be saved. If they are to be saved, a file name will be generated that consists of the following elements separated by full stops: the response variable name with .back if backtransformed values are being plotted, the classify term, Bar or Line and, if error.intervals is not "none", one of SE, CI or LSI. The file will be saved as a 'png' file in the current work directory.

transform.power — A numeric specifying the power of a transformation, if one has been applied to the response variable. Unless it is equal to 1, the default, back-transforms of the predictions will be obtained and stored in the backtransforms component of the alldiffs.object. The plots and tables arguments control the plotting and output of the predictions and backtransforms. The back-transformation raises the predictions to the power equal to the reciprocal of transform.power, unless it equals 0 in which case the exponential of the predictions is taken.

offset — A number that has been added to each value of the response after any scaling and before applying any power transformation. Unless it is equal to 0, the default, back-transforms of the predictions will be obtained and stored in the backtransforms component of the alldiffs.object. The plots and tables arguments control the plotting and output of the predictions and backtransforms. The backtransformation will, after backtransforming for any power transformation, subtract the offset.

scale — A number by which each value of the response has been multiply before adding any offset and applying any power transformation. Unless it is equal to 1, the default, back-transforms of the predictions will be obtained and stored in the backtransforms component of the alldiffs.object. The plots and tables arguments control the plotting and output of the predictions and backtransforms. The backtransformation will, after backtransforming for any power transformation and then subtracting the offset, divide by the scale.

transform.function — A character giving the name of a function that specifies the scale on which the predicted values are defined. This may be the result of a transformation of the data using the function or the use of the function as a link function in the fitting of a generalized linear (mixed) model (GL(M)M). The possible transform.functions are identity, log, inverse, sqrt, logit, probit, and cloglog. The predicted.values and error.intervals, if not StandardError intervals, will be back-transformed using the inverse function of the transform.function. The standard.error column will be set to NA, unless (i) asreml returns columns named transformed.value and approx.se, as well as those called predicted.values and standard.error (such as when a GLM is fitted) and (ii) the values in transformed.value are equal to those obtained by backtransforming the predicted.values

using the inverse function of the transform.function. Then, the approx.se values will be saved in the standard.error column of the backtransforms component of the returned alldiffs.obj. Also, the transformed.value and approx.se columns are removed from both the predictions and backtransforms components of the alldiffs.obj. Note that the values that end up in the standard errors column are approximate for the backtransformed values and are not used in calculating error.intervals.

tables — A character vector containing a combination of predictions, vcov, backtransforms, differences, p.differences, sed, LSD and all. These nominate which components of the alldiffs.object to print.

level.length — The maximum number of characters from the levels of factors to use in the row and column labels of the tables produced by allDifferences.data.frame.

sortFactor — A character containing the name of the factor that indexes the set of predicted values that determines the sorting of the components. If there is only one variable in the classify term then sortFactor can be NULL and the order is defined by the complete set of predicted values. If there is more than one variable in the classify term then sortFactor must be set. In this case the sortFactor is sorted in the same order within each combination of the values of the sortParallelToCombo variables: the classify variables, excluding the sortFactor. There should be only one predicted value for each unique value of sortFactor within each set defined by a combination of the values of the classify variables, excluding the sortFactor factor. The order to use is determined by either sortParallelToCombo or sortOrder.

sortParallelToCombo

A list that specifies a combination of the values of the factors and numerics, excluding sortFactor, that are in classify. Each of the components of the supplied list is named for a classify variable and specifies a single value for it. The combination of this set of values will be used to define a subset of the predicted values whose order will define the order of sortFactor. Each of the other combinations of the values of the factors and numerics will be sorted in parallel. If sortParallelToCombo is NULL then the first value of each classify variable, except for the sortFactor factor, in the predictions component is used to define sortParallelToCombo. If there is only one variable in the classify then sortParallelToCombo is ignored.

sortNestingFactor

A character containing the name of the factor that defines groups of the sortFactor within which the predicted values are to be ordered. If there is only one variable in the classify then sortNestingFactor is ignored.

sortOrder — A character vector whose length is the same as the number of levels for sortFactor in the predictions component of the alldiffs.object. It specifies the desired order of the levels in the reordered components of the alldiffs.object. The argument sortParallelToCombo is ignored.

The following creates a sortOrder vector levs for factor f based on the values in x: levs <- levels(f)[order(x)].

decreasing — A logical passed to order that detemines whether the order for sorting the components of the alldiffs.object is for increasing or decreasing magnitude of the predicted values.

trace	If TRUE then partial iteration details are displayed when ASReml-R functions are invoked; if FALSE then no output is displayed.
ggplotFuncs	A list, each element of which contains the results of evaluating a ggplot2 function. It is created by calling the list function with a ggplot2 function call for each element. It is passed to plotPredictions.data.frame.
...	further arguments passed to predict.asreml via predictPlus.asreml and to ggplot via plotPredictions.data.frame.

Value

A list containing an alldiffs.object for each term for which tables are produced. The names of the components of this list are the terms with full-stops (.) replacing colons (:). Plots are also preduced depending on the setting of the plot argument.

Author(s)

Chris Brien

See Also

predictPlus.asreml, allDifferences.data.frame, sort.alldiffs, subset.alldiffs, redoErrorIntervals.alldiffs, recalcLSD.alldiffs, pickLSDstatistics.alldiffs, plotPredictions.data.frame, print.alldiffs, as.Date, Devices

Examples

```
## Not run:
data(WaterRunoff.dat)
titles <- list("Days since first observation", "Days since first observation",
               "pH", "Turbidity (NTU)")
names(titles) <- names(WaterRunoff.dat)[c(5,7,11:12)]
asreml.options(keep.order = TRUE) #required for asreml-R4 only
current.asr <- asreml(fixed = log.Turbidity ~ Benches + Sources + Type + Species +
                                Sources:Type + Sources:Species + Sources:Species:xDay +
                                Sources:Species:Date,
                      data = WaterRunoff.dat, keep.order = TRUE)
current.asrt <- as.asrtests(current.asr, NULL, NULL)

#### Get the observed combinations of the factors and variables in classify
class.facs <- c("Sources","Species","Date","xDay")
levs <- as.data.frame(table(WaterRunoff.dat[class.facs]))
levs <- levs[do.call(order, levs), ]
levs <- as.list(levs[levs$Freq != 0, class.facs])
levs$xDay <- as.numfac(levs$xDay)

#### parallel and levels are arguments from predict.asreml
diff.list <- predictPresent.asreml(asreml.obj = current.asrt$asreml.obj,
                                   terms = "Date:Sources:Species:xDay",
                                   x.num = "xDay", x.fac = "Date",
                                   parallel = TRUE, levels = levs,
                                   wald.tab = current.asrt$wald.tab,
```

```
                                    plots = "predictions",
                                    error.intervals = "StandardError",
                                    titles = titles,
                                    transform.power = 0,
                                    present = c("Type","Species","Sources"),
                                    tables = "none",
                                    level.length = 6)

## End(Not run)
```

`print.alldiffs` *Prints the values in an* `alldiffs.object` *in a nice format.*

Description

Prints the predictions and standard errors from a fitted model, including the attributes of the `predictions.frame`. Also prints out all pairwise differences between the predictions to 2 significant figures, along with their p-values and standard errors to 4 decimal places. If LSDs are requested the mean, minimum and maximum LSDs will be printed.

Usage

```
## S3 method for class 'alldiffs'
print(x, which = "all", colourise = FALSE, ...)
```

Arguments

`x`	An `alldiffs.object`.
`which`	A `character vector` containing a combination of `predictions`, `vcov`, `backtransforms`, `differences`, `p.differences`, `sed`, `LSD` and `all`. These nominate which components of the `alldiffs.object` to print.
`colourise`	A `logical` which, if `TRUE`, results in the header text produced by `predict.asreml` being displayed in a different colour, if supported by the output terminal device. It overrides the `TRUE` setting of the `colourise` argument of `asreml::asreml.options`.
`...`	further arguments passed to `print.predictions.frame`.

Value

No value is returned, but the components of `x` are printed.

Author(s)

Chris Brien

See Also

`print.predictions.frame`, `as.alldiffs`, `allDifferences.data.frame`

Examples

```
## Not run:
print.alldiffs(diffs, which = "predictions")

## End(Not run)
```

`print.asrtests` *Prints the values in an* `asrtests.object`

Description

Prints a `summary` of the asreml object, the `pseudoanova` and the test.summary `data.frame` that are stored in the `asrtests.object`.

Usage

```
## S3 method for class 'asrtests'
print(x, which = "key", colourise = FALSE, ...)
```

Arguments

`x`	An `asrtests.object`.
`which`	Which elements of the `asrtests.object` to print. Possible values are some combination of `asremlsummary`, `vparametersummary`, `pseudoanova`, `wald.tab`, `testsummary` and `key` or `all`. The option `wald.tab` is a synonym for `pseudoanova`. The options `key` and `all` are mutually exclusive; `key` includes `vparametersummary`, but not the rest of `asremlsummary`, while `all` includes the full `asremlsummary` that includes the `vparametersummary`.
`colourise`	A `logical` which, if `TRUE`, results in the header text produced by `wald.asreml` being displayed in a different colour, if supported by the output terminal device. It overrides the `TRUE` setting of the `colourise` argument of `asreml::asreml.options`.
`...`	further arguments passed to `print` and `print.wald.tab`.

Value

No value is returned, but the elements of the list in `x` are printed.

Author(s)

Chris Brien

See Also

`print.wald.tab`, `as.asrtests`, `asremlPlus-package`

Examples

```
## Not run:
data(Wheat.dat)
current.asr <- asreml(yield ~ Rep + WithinColPairs + Variety,
                      random = ~ Row + Column + units,
                      residual = ~ ar1(Row):ar1(Column),
                      data=Wheat.dat)
current.asrt <- as.asrtests(current.asr, NULL, NULL)
current.asrt <- rmboundary.asrtests(current.asrt)
# Test Row autocorrelation
current.asrt <- testresidual(current.asrt, , "~ Row:ar1(Column)",
                             label="Row autocorrelation", simpler=TRUE)
print(current.asrt)

## End(Not run)
```

`print.LSDdata` *Prints the components of a list containing data on the LSDs for all pairwise differences of predictions.*

Description

Prints the components of an `LSDdata` `list` created by `exploreLSDs`, that contains data on the LSDs for all pairwise differences of predictions stored in an `alldiffs.object`.

Usage

```
## S3 method for class 'LSDdata'
print(x, which.print = c("statistics", "false.pos", "false.neg"), ...)
```

Arguments

`x`	An object that, ideally, is of class `LSDdata`.
`which.print`	Which components of the `LSDdata` `list` to print. Possible values are any combination of `frequencies`, `distinct.vals`, `statistics`, `accuracy`, `false.pos`, `false.neg`, `per.pred.accuracy`, `LSD`, `summary` and `all`, except that `summary` and `all` cannot occur together. For a description of the components, see `alldiffs.object`. The default is to print `statistics`, `false.pos`, `false.neg`. The option `summary` results in the printing of `distinct.vals`, `statistics`, `false.pos`, `false.neg`.
`...`	further arguments passed to `print`.

Value

No value is returned, but components of `x` are printed as specified in `which.print`.

Author(s)

Chris Brien

See Also

exploreLSDs.alldiffs, alldiffs.object

Examples

```
## Not run:
data(WaterRunoff.dat)
asreml.options(keep.order = TRUE) #required for asreml-R4 only
current.asr <- asreml(fixed = pH ~ Benches + (Sources * (Type + Species)),
                      random = ~ Benches:MainPlots,
                      keep.order=TRUE, data= WaterRunoff.dat)
current.asrt <- as.asrtests(current.asr, NULL, NULL)
diffs <- predictPlus(classify = "Sources:Type",
                     asreml.obj = current.asr,
                     wald.tab = current.asrt$wald.tab,
                     present = c("Sources", "Type", "Species"))
LSDdata <- exploreLSDs(diffs, LSDtype = "factor.combinations", LSDby = "Sources")
print(LSDdata)

## End(Not run)
```

---

print.predictions.frame

Prints the values in a predictions.frame*, with or without title and heading.*

Description

Prints the predictions from a fitted model, along with their standard errors and, if present, their error intervals, with or without title and headings.

Usage

```
## S3 method for class 'predictions.frame'
print(x, title = NULL,
      which.predictions = c("title", "heading", "table"),
      colourise = FALSE, ...)
```

Arguments

x	An object that, ideally, is of class predictions.frame.
title	A character giving a title to be printed out before the heading and table for the predictions.frame,
which.predictions	what Which elements of the predictions.frame to print. Possible values are some combination of title, heading, table and all. The heading is an attribute of x.

colourise	A logical which, if TRUE, results in the header text produced by predict.asreml being displayed in a different colour, if supported by the output terminal device. It overrides the TRUE setting of the colourise argument of asreml::asreml.options, but is only operational when the table is also printed.
...	further arguments passed to print.predictions.frame.

Value

No value is returned, but the components of x are printed.

Author(s)

Chris Brien

See Also

print.alldiffs, as.alldiffs, allDifferences.data.frame

Examples

```
## Not run:
print.predictions.frame(diffs$predictions, which = "all")

## End(Not run)
```

---

print.test.summary *Prints a data.frame containing a test.summary.*

Description

Prints a test.summary (also a choose.summary) with or without a title and with p-values limited to 4-digits.

Usage

```
## S3 method for class 'test.summary'
print(x, which.print = c("title", "table"),
      omit.columns = NULL, response = NULL, ...)
```

Arguments

x	A object that, ideally, is of class test.summary.
which.print	A character specifying the aspects of the test.summary to print. Possible values are some combination of title, table and all.
omit.columns	A character specifying the columns of the test.summary table to be omitted from the print. If NULL, none are omitted.

response	A character specifying the name of the response that the test.summary table is based on and is to be printed in the heading for the table. If NULL, no response name is printed.
...	further arguments passed to print, but is only operational when the table is also printed.

Value

No value is returned, but x is printed, possibly with a title.

Author(s)

Chris Brien

See Also

print.wald.tab, print.asrtests, as.asrtests, asremlPlus-package

Examples

```
## Not run:
data(Wheat.dat)
current.asr <- asreml(yield ~ Rep + WithinColPairs + Variety,
                      random = ~ Row + Column + units,
                      residual = ~ ar1(Row):ar1(Column),
                      data=Wheat.dat)
current.asrt <- as.asrtests(current.asr, NULL, NULL)
current.asrt <- rmboundary.asrtests(current.asrt)
# Test Row autocorrelation
current.asrt <- testresidual(current.asrt, "~ Row:ar1(Column)",
                             label="Row autocorrelation", simpler=TRUE)
print(current.asrt$test.summary)

## End(Not run)
```

---

print.wald.tab *Prints a data.frame containing a Wald or pseudoanova table.*

Description

Prints a wald.tab with or without title and/or heading. The printing of the p-values is limited to 4 digits.

Usage

```
## S3 method for class 'wald.tab'
print(x, which.wald = c("title", "heading", "table"),
      colourise = FALSE, ...)
```

Arguments

x	An object that, ideally, is of class wald.tab.
which.wald	Which elements of the wald.tab to print. Possible values are some combination of title, heading, table and all. The heading is an attribute of x.
colourise	A logical which, if TRUE, results in the header text produced by wald.asreml being displayed in a different colour, if supported by the output terminal device. It overrides the TRUE setting of the colourise argument of asreml::asreml.options.
...	further arguments passed to print and print.wald.tab, but is only operational when the table is also printed.

Value

No value is returned, but x is printed as specified in which.wald.

Author(s)

Chris Brien

See Also

print.test.summary, print.asrtests, as.asrtests, asremlPlus-package

Examples

```
## Not run:
data(Wheat.dat)
current.asr <- asreml(yield ~ Rep + WithinColPairs + Variety,
                      random = ~ Row + Column + units,
                      residual = ~ ar1(Row):ar1(Column),
                      data=Wheat.dat)
current.asrt <- as.asrtests(current.asr, NULL, NULL)
current.asrt <- rmboundary.asrtests(current.asrt)
# Test Row autocorrelation
current.asrt <- testresidual(current.asrt, "~ Row:ar1(Column)",
                             label="Row autocorrelation", simpler=TRUE)
print(current.asrt$wald.tab)

## End(Not run)
```

---

printFormulae.asreml *Prints the formulae from an asreml object.*

Description

Prints the formulae nominated in the which argument from the call stored in an asreml object. The output is broken into lines with print width equal to the option width.

Usage

```
## S3 method for class 'asreml'
printFormulae(asreml.obj, which = c("fixed", "random", "residual"),
              expanded = FALSE, envir = parent.frame(), ...)
```

Arguments

asreml.obj	An asreml object resulting from the fitting of a model using REML.
which	A character listing the formula(e) to be printed from the call stored in asreml.obj. It should be some combination of fixed, random, residual, sparse and all. If all is included then all formula(e) will be printed.
expanded	A logical indicating whether terms are to be expanded to the sum of a set of individual terms.
envir	The environment in which the formula(e) are to be evaluated. May also be NULL, a list, a data.frame, a pairlist or an integer as specified to sys.call.
...	Arguments passed on to getFormulae.asreml and ultimately to update.formula and terms.formula.

Value

Invisibly returns a character, each element of which contains one of the extracted formulae.

Author(s)

Chris Brien

See Also

printFormulae.asreml

Examples

```
## Not run:
   data(Wheat.dat)
   current.asr <- asreml(yield ~ Rep + WithinColPairs + Variety,
                         random = ~ Row + Column + units,
                         residual = ~ ar1(Row):ar1(Column),
                         data=Wheat.dat)
    printFormulae(current.asr)

## End(Not run)
```

`R2adj.asreml` *Calculates the adjusted coefficient of determination for a specified combination of fixed and random terms.*

Description

Calculates the adjusted coefficient of determination (R2) that measures the contributions to the total variance exhibited by the observations of a specified combination of fixed and random terms in a fitted linear mixed model.

Note that the adjusted R2 can be negative, which indicates that the contribution of the terms to the total variance is very small relative to the sum of the degrees of freedom of the terms.

Piepho's (2023) method for GLMMs has not been implemented. This function is not available for ASReml-R version 3.

Usage

```
## S3 method for class 'asreml'
R2adj(asreml.obj,
      include.which.fixed = ~ ., orthogonalize = "hybrid",
      include.which.random = NULL,
      bound.exclusions = c("F","B","S","C"), ...)
```

Arguments

`asreml.obj` An `asreml` object returned from a call to `asreml`.

`include.which.fixed`

A `formula` specifying the fixed terms whose joint contribution to the total variance is to be measured. If it is `NULL`, no fixed term is to be included in the terms whose joint contribution is to be assessed. The `formula` `~ .` indicates that the joint contribution of all fixed terms are to be measured. Otherwise, the joint contribution of the set of terms specified by the `formula` will be assessed. The `formula` can include a `"."`, which means all fixed terms currently fitted, and is most likely followed by a `"-"` with a bracketed set of terms to be removed that can be specified using `formula` operators. The names of the resulting terms must be the same as those in either the `terms` attribute of the `fixed` component of the `coefficient` component of the supplied `asreml.obj`, or the Wald table produced by `wald.asreml`.

Note that the contribution of a subset of the fixed terms is only unique if the effects for the fixed terms are orthogonal; if the effects are not orthogonal then the contributions will depend on the order of the terms in the formula. Also, determining the joint contribution of a subset of the fixed terms in the model may be computationally demanding because the projection matrices have to be formed for all fixed terms and these projections matrices have to be orthogonalized. A heavy computational burden is most likely when the effects for the fixed terms are not orthogonal, for example, when numeric covariates are included amongst the terms.

`orthogonalize` A `character` vector indicating the method for orthogonalizing a projector to those for terms that occurred previously in the `formula` for `include.which.fixed`. Orthogonalizing the projectors of fixed terms is not performed for the default setting of `. ~`. WHen required, two options are available for orthogonalizing: `hybrid` and `eigenmethods`. The `hybrid` option is the most general and uses the relationships between the projection operators for the terms in the `formula` to decide which `projectors` to subtract and which to orthogonalize using eigenmethods. The `eigenmethods` option recursively orthogonalizes the `projectors` using an eigenanalysis of each `projector` with previously orthogonalized `projectors`. See the documentation for `porthogonalize.list` from the R package dae for more information.

`include.which.random`
A `formula` specifying the random terms whose joint contribution to the total variance is to be measured. If it is `NULL`, no random term is to be included in the terms whose joint contribution is to be assessed. The formula `~ .` indicates that the joint contribution of all random terms is to be measured. Otherwise, the joint contribution of the set of terms specified by the `formula` will be assessed. The `formula` can include a `"."`, which means all random terms currently fitted, and is most likely followed by a `"-"` with a bracketed set of terms to be removed that can be specified using `formula` operators. The resulting terms must be one of those occurring in either the `vparameters` component of the supplied `asreml.obj`, or in the `terms` attribute of the `random` component of the `coefficient` component of the supplied `asreml.obj`.

`bound.exclusions`
A `character` specifying one or more bound codes that will result in a variance parameter in the `random` model being excluded from contributing to the variance. If set to `NULL` then none will be excluded.

`...` Provision for passing arguments to functions called internally - not used at present.

Details

The method used to compute the adjusted R2 under a linear mixed model (LMM) is that described by Piepho (2023). Here, the method has been extended to allow computation of the adjusted R2 for a subset of the fixed terms. A set of orthogonalized projectors for all of the fixed terms in the model (a set of $\mathbf{Q}_i$s) is obtained and the combined contribution of the fixed terms nominated in `include.which.fixed` is obtained by computing the average semisquared bias, ASSB, for the nominated fixed terms as:

$$\Sigma_i\{(\mathbf{Q}_i\mathbf{X}\boldsymbol{\beta})^{\mathrm{T}}\mathbf{Q}_i\mathbf{X}\boldsymbol{\beta} + \mathrm{trace}(\mathbf{X}^{\mathrm{T}}\mathbf{Q}_i\mathbf{X}\mathrm{var}(\boldsymbol{\beta}))\}/(n-1)$$

Of the two methods, `eigenmethods` is least likely to fail, but it does not establish the marginality between the terms. It is often needed when there is nonorthogonality between terms, such as when there are several linear covariates. It can also be more efficient in these circumstances.

The process can be computationally expensive, particularly for a large data set (500 or more observations) and/or when many terms are to be orthogonalized, particularly if they are not orthogonal.

If the error `"Matrix is not idempotent"` should occur then, especially if there are many terms, one might try using `set.daeTolerance` from the dae package to reduce the tolerance used in

determining if values are either the same or are zero; it may be necessary to lower the tolerance to as low as 0.001. Also, setting `orthogonalize` to `eigenmethods` is worth a try.

In doing the computations, no changes are made to the fitted model, nor is the `formula` stored in `asreml.obj` referred to. Instead, the names of the terms referred to are those stored in the `coefficients` component of the `asreml.obj`. Use `attr(asreml.obj$coefficients$fixed, which = "terms")` to access the attribute for fixed terms; substitute `random` for `fixed` to see the names of the random terms. For fixed terms. the term names are the same as those in the Wald table produced by `wald.asreml`, and, for random terms, the same as those in the `vparameters` component of the `asreml.obj`. Two `asreml` `formula` functions whose terms can differ from their formulation in a model `formula` are `at` and `str`.)

The function `estimateV.asreml` is used to calculate the variance matrices required in calculating the adjusted R2.

Value

A `numeric` that is the adjusted R2, expressed as a percentage. It has attributes `include.which.fixed`, `include.which.random` and `missing.termmatrix` (use `attr(x, which = "name")` to access the attribute `name`). The `missing.termmatrix` attribute will be `NULL`, unless the design matrix could not be obtained for one or more model terms. If is is not `NULL`, it will be a list of terms whose design matices could not be produced and so are not included in the variance matrix estimate. An `NA` will be returned for the adjusted R2 if `missing.termmatrix` is not `NULL` or a generalized inverse could not be computed for the variance matrix estimate.

Author(s)

Chris Brien

References

Piepho, H.-P. (2023). An adjusted coefficient of determination (R2) for generalized linear mixed models in one go. *Biometrical Journal*, **65(7)**, 2200290. doi:10.1002/bimj.202200290.

See Also

`asreml`, `estimateV.asreml`.

Examples

```
## Not run:
  data(Oats.dat)

  current.asr <- asreml(Yield ~ Nitrogen*Variety,
                        random=~Blocks/Wplots,
                        data=Oats.dat)
  R2.adj.fix <- R2adj.asreml(current.asr)
  R2.adj.ran <- R2adj.asreml(current.asr,
                             include.which.fixed = NULL, include.which.random = ~ .)
  R2.adj.tot <- R2adj.asreml(current.asr, include.which.random = ~ .)
  R2.adj.tot <- R2adj.asreml(current.asr, include.which.random = ~ Blocks)
  R2.adj.add <- R2adj.asreml(current.asr, include.which.fixed = ~ Nitrogen + Variety)
```

```
  R2.adj.int <- R2adj.asreml(current.asr,
                             include.which.fixed = ~ . - (Nitrogen + Variety))
  R2.adj.int <- R2adj.asreml(current.asr, include.which.fixed = ~ Nitrogen:Variety)

## End(Not run)
```

`ratioTransform.alldiffs` *Calculates the ratios of nominated pairs of predictions stored in an* `alldiffs.object`.

Description

Ratio predictions and error intervals are formed for two levels of a factor, the `ratio.factor`. For each pair of a level of the `ratio.factor` in `numerator.levels` with a level in `denominator.levels`, the ratio predictions are formed from all combinations of the other factors as the ratio of the two predictions for each combination, along with confidence intervals for the ratio predictions computed using the Fieller (1954) method.

The printing of the components produced is controlled by the `tables` argument.

Usage

```
## S3 method for class 'alldiffs'
ratioTransform(alldiffs.obj, ratio.factor,
               numerator.levels, denominator.levels,
               method = "Fieller", alpha = 0.05,
               response = NULL, response.title = NULL,
               tables = "predictions", ...)
```

Arguments

`alldiffs.obj`	An `alldiffs.object`.
`ratio.factor`	A `character` string giving the name of the factor for whose levels the ratios are to be calculated.
`numerator.levels`	A `character` string containing the levels of `ratio.factor` to be used as numerators of the ratio.
`denominator.levels`	A `character` string containing the levels of `ratio.factor` to be used as denominators of the ratio.
`method`	A `character` string specifying the method to use in calculating the ratios and their `error.intervals`. At present only `Fieller` is available. For the `Fieller` `method`, ratios of predictions are formed and confidence intervals formed for them using Fieller's (1954) theorem.
`alpha`	A `numeric` giving the significance level for LSDs or one minus the confidence level for confidence intervals.

response	A character specifying the response variable for the predictions. It is stored as an attribute to the alldiffs.object .
response.title	A character specifying the title for the response variable for the predictions. It is stored as an attribute to the alldiffs.object.
tables	A character vector containing either none or predictions
...	further arguments passed to linTransform.alldiffs.

Value

A list of predictions.frames, each containing the ratio predictions and their confidence limits for a combination of the numerator.levels with the denominator.levels. It will also contain the values of the variables in the classify of alldiffs.obj that index the ratio predictions, except that the ratio.factor is omitted.

If sortFactor attribute of the alldiffs.object is set and is not the ratio.factor, the predictions and their backtransforms will be sorted using the sortOrder attribute of the alldiffs.object.

Author(s)

Chris Brien

References

Fieller, E. C. (1954). Some Problems in Interval Estimation. *Journal of the Royal Statistical Society.Series B (Methodological)*, **16**, 175-185.

See Also

pairdiffsTransform, linTransform, predictPlus.asreml, as.alldiffs, print.alldiffs, sort.alldiffs, subset.alldiffs, allDifferences.data.frame, redoErrorIntervals.alldiffs, recalcLSD.alldiffs, predictPresent.asreml, plotPredictions.data.frame, as.Date, predict.asreml

Examples

```
#### Form the ratios and Fieller CIs for RGR Salinity
load(system.file("extdata", "testDiffs.rda", package = "asremlPlus", mustWork = TRUE))
Preds.ratio.RGR <- ratioTransform(diffs.RGR,
                                  ratio.factor = "Salinity",
                                  numerator.levels = "Salt",
                                  denominator.levels = "Control")

#### Form the ratios and Fieller CIs for Nitrogen compared to no Nitrogen
data("Oats.dat")
## Not run:
m1.asr <- asreml(Yield ~ Nitrogen*Variety,
                 random=~Blocks/Wplots,
                 data=Oats.dat)
current.asrt <- as.asrtests(m1.asr)
wald.tab <-  current.asrt$wald.tab
```

```
Var.diffs <- predictPlus(m1.asr, classify="Nitrogen:Variety", pairwise = TRUE,
                         Vmatrix = TRUE, error.intervals = "halfLeast",
                         LSDtype = "factor", LSDby = "Variety",
                         wald.tab = wald.tab)

## End(Not run)

 ## Use lme4 and emmmeans to get predictions and associated statistics
if (requireNamespace("lmerTest", quietly = TRUE) &
    requireNamespace("emmeans", quietly = TRUE))
{
  m1.lmer <- lmerTest::lmer(Yield ~ Nitrogen*Variety + (1|Blocks/Wplots),
                            data=Oats.dat)
  ## Set up a wald.tab
  int <- as.data.frame(rbind(rep(NA,4)))
  rownames(int) <- "(Intercept)"
  wald.tab <- anova(m1.lmer, ddf = "Kenward", type = 1)[,3:6]
  names(wald.tab) <- names(int) <- c("Df", "denDF", "F.inc", "Pr")
  wald.tab <- rbind(int, wald.tab)
  #Get predictions
  Var.emm <- emmeans::emmeans(m1.lmer, specs = ~ Nitrogen:Variety)
  Var.preds <- summary(Var.emm)
  ## Modify Var.preds to be compatible with a predictions.frame
  Var.preds <- as.predictions.frame(Var.preds, predictions = "emmean",
                                    se = "SE", interval.type = "CI",
                                    interval.names = c("lower.CL", "upper.CL"))
  Var.vcov <- vcov(Var.emm)
  Var.sed <- NULL
  den.df <- wald.tab[match("Variety", rownames(wald.tab)), "denDF"]

  #Create alldiffs object
  Var.diffs <- as.alldiffs(predictions = Var.preds,
                           sed = Var.sed, vcov = Var.vcov,
                         classify = "Nitrogen:Variety", response = "Yield", tdf = den.df)
}

if (exists("Var.diffs"))
  Preds.ratio.OatsN <- ratioTransform(alldiffs.obj = Var.diffs,
                                      ratio.factor = "Nitrogen",
                                      numerator.levels = c("0.2","0.4","0.6"),
                                      denominator.levels = "0.2")
```

---

recalcLSD.alldiffs *Adds or recalculates the* LSD.frame *that is a component of an* alldiffs.object.

Description

Given an alldiffs.object, adds or recalculate its LSD.frame. **N.B. No changes are made to the error.intervals — use** redoErrorIntervals.alldiffs **to modify both the error.intervals and the** LSD.frame**.**

Usage

```
## S3 method for class 'alldiffs'
recalcLSD(alldiffs.obj, LSDtype = "overall", LSDsupplied = NULL,
          LSDby = NULL, LSDstatistic = "mean", LSDaccuracy = "maxAbsDeviation",
          alpha = 0.05, ...)
```

Arguments

alldiffs.obj	An alldiffs.object.
LSDtype	A character string that can be overall, factor.combinations, per.prediction or supplied. It determines whether the values stored in a row of a LSD.frame are the values calculated (i) overall from the LSD values for all pairwise comparisons, unless there is only one, possibly repeated, prediction, when a notional LSD is calculated, (ii) the values calculated from the pairwise LSDs for the levels of each factor.combination, unless there is only one prediction for a level of the factor.combination, when a notional LSD is calculated, (iii) per.prediction, being based, for each prediction, on all pairwise differences involving that prediction, or (iv) as supplied values of the LSD, specified with the LSDsupplied argument; these supplied values are to be placed in the assignedLSD column of the LSD.frame stored in an alldiffs.object so that they can be used in LSD calculations. See LSD.frame for further information on the values in a row of this data.frame and how they are calculated.
LSDsupplied	A data.frame or a named numeric containing a set of LSD values that correspond to the observed combinations of the values of the LSDby variables in the predictions.frame or a single LSD value that is an overall LSD. If a data.frame, it may have (i) a column for the LSDby variable and a column of LSD values or (ii) a single column of LSD values with rownames being the combinations of the observed values of the LSDby variables. Any name can be used for the column of LSD values; assignedLSD is sensible, but not obligatory. Otherwise, a numeric containing the LSD values, each of which is named for the observed combination of the values of the LSDby variables to which it corresponds. (Applying the function dae::fac.combine to the predictions component is one way of forming the required combinations for the (row) names.) The values supplied will be incorporated into assignedLSD column of the LSD.frame stored as the LSD component of the alldiffs.object.
LSDby	A character (vector) of variables names, being the names of the factors or numerics in the classify; for each combination of their levels and values, there will be or is a row in the LSD.frame stored in the LSD component of the alldiffs.object when LSDtype is factor.combinatons.
LSDstatistic	A character nominating one or more of minimum, q10, q25, mean, median, q75, q90 or maximum as the value(s) to be stored in the assignedLSD column in an LSD.frame; the values in the assignedLSD column are used in computing halfLeastSignificant error.intervals. Here q10, q25, q75 and q90 indicate the sample quantiles corresponding to probabilities of 0.1, 0.25, 0.75 and 0.9 for the group of LSDs from which a single LSD value is calculated. The function quantile is used to obtain them. The mean LSD is calculated as the

square root of the mean of the squares of the LSDs for the group. The median is calculated using the median function. Multiple values are only produced for LSDtype set to factor.combination, in which case LSDby must not be NULL and the number of values must equal the number of observed combinations of the values of the variables specified by LSDby. If LSDstatistic is NULL, it is reset to mean.

LSDaccuracy — A character nominating one of maxAbsDeviation, maxDeviation, q90Deviation or RootMeanSqDeviation as the statistic to be calculated as a measure of the accuracy of assignedLSD. The option q90Deviation produces the sample quantile corresponding to a probability of 0.90. The deviations are the differences between the LSDs used in calculating the LSD statistics and each assigned LSD and the accuracy is expressed as a proportion of the assigned LSD value. The calculated values are stored in the column named accuracyLSD in an LSD.frame.

alpha — The significance level for an LSD to compare a pair of predictions. It is stored as an attribute to the alldiffs.object.

... — further arguments passed to allDifferences.data.frame; attributes transform.power, offset and scale cannot be passed.

Value

An alldiffs.object with components predictions, vcov, differences, p.differences sed, LSD and, if present in alldiffs.obj, backtransforms.

Author(s)

Chris Brien

See Also

asremlPlus-package, as.alldiffs, sort.alldiffs, subset.alldiffs, print.alldiffs, renewClassify.alldiffs, exploreLSDs.alldiffs, pickLSDstatistics.alldiffs, redoErrorIntervals.alldiffs, plotPredictions.data.frame, predictPlus.asreml, predictPresent.asreml

Examples

```
data(WaterRunoff.dat)

##Use asreml to get predictions and associated statistics

## Not run:
asreml.options(keep.order = TRUE) #required for asreml-R4 only
current.asr <- asreml(fixed = pH ~ Benches + (Sources * (Type + Species)),
                      random = ~ Benches:MainPlots,
                      keep.order=TRUE, data= WaterRunoff.dat)
current.asrt <- as.asrtests(current.asr, NULL, NULL)
TS.diffs <- predictPlus(classify = "Sources:Type",
                        asreml.obj = current.asr,
                        wald.tab = current.asrt$wald.tab,
                        present = c("Sources", "Type", "Species"))
```

```
## End(Not run)

## Use lmeTest and emmmeans to get predictions and associated statistics

if (requireNamespace("lmerTest", quietly = TRUE) &
    requireNamespace("emmeans", quietly = TRUE))
{
  m1.lmer <- lmerTest::lmer(pH ~ Benches + (Sources * (Type + Species)) +
                              (1|Benches:MainPlots),
                            data=na.omit(WaterRunoff.dat))
  TS.emm <- emmeans::emmeans(m1.lmer, specs = ~ Sources:Species)
  TS.preds <- summary(TS.emm)
  den.df <- min(TS.preds$df, na.rm = TRUE)
  ## Modify TS.preds to be compatible with a predictions.frame
  TS.preds <- as.predictions.frame(TS.preds, predictions = "emmean",
                                   se = "SE", interval.type = "CI",
                                   interval.names = c("lower.CL", "upper.CL"))

  ## Form an all.diffs object and check its validity
  TS.vcov <- vcov(TS.emm)
  TS.diffs <- allDifferences(predictions = TS.preds, classify = "Sources:Species",
                             vcov = TS.vcov, tdf = den.df)
  validAlldiffs(TS.diffs)
}

## Plot p-values for predictions obtained using asreml or lmerTest
if (exists("TS.diffs"))
{
  ##Recalculate the LSD values for predictions obtained using asreml or lmerTest
  TS.diffs <- recalcLSD.alldiffs(TS.diffs, LSDtype = "factor.combinations",
                                 LSDby = "Sources")
}
```

`recalcWaldTab.asrtests` *Recalculates the denDF, F.inc and P values for a table of Wald test statistics obtained using* `wald.asreml`

Description

If some or all denDF are not available, either because they are NA or because F.inc values were not calculated, this function allows the user to specify how approximate denDF values are to be obtained. This is done through the `dDF.fault` and `dDF.values` arguments. Note that if denDF values are available in the Wald table then only those that are `NA` will be replaced. The P values are recalculated using F.con, if present in the `wald.tab`, otherwise F.inc is used. It is noted that, as of asreml version 4, `wald.asreml` has a `kenadj` argument.

Usage

```
## S3 method for class 'asrtests'
recalcWaldTab(asrtests.obj, recalc.wald = FALSE,
              denDF="numeric", dDF.fault = "none",
              dDF.values = NULL, trace = FALSE, ...)
```

Arguments

asrtests.obj	an asrtests.object containing the components (i) asreml.obj, (ii) wald.tab, and (iii) test.summary.
recalc.wald	A logical indicating whether to call wald.asreml to recalculate the pseudoanova table for the model fit stored in the asreml object contained in asrtests.obj.
denDF	Specifies the method to use in computing approximate denominator degrees of freedom when wald.asreml is called. Can be none to suppress the computations, numeric for numerical methods, algebraic for algebraic methods or default, the default, to automatically choose numeric or algebraic computations depending on problem size. The denominator degrees of freedom are calculated according to Kenward and Roger (1997) for fixed terms in the dense part of the model.
dDF.fault	A character specifying the method to use to obtain substitute denominator degrees of freedom. when the numeric or algebraic methods produce faulty values, viz. NA, Inf or less than 0.01. Consistent with when no denDF are available, the default is "residual" and so the residual degrees of freedom from asreml.obj$nedf are used. If dDF.fault = "none", no substitute denominator degrees of freedom are employed; if dDF.fault = "residual"; if dDF.fault = "maximum", the maximum of those denDF that are available, excluding that for the Intercept, is used; if all denDF are faulty, asreml.obj$nedf is used. If dDF.fault = "supplied", a vector of values for the denominator degrees of freedom is to be supplied in dDF.values. Any other setting is ignored and a warning message produced. Generally, substituting these degrees of freedom is anticonservative in that it is likely that the degrees of freedom used will be too large.
dDF.values	A vector of values to be used when dDF.fault = "supplied". Its values will be used when denDF in a test for a fixed effect is NA. This vector must be the same length as the number of fixed terms, including (Intercept) whose value could be NA.
trace	If TRUE then partial iteration details are displayed when ASReml-R functions are invoked; if FALSE then no output is displayed.
...	further arguments passed to asreml and to wald.asreml.

Value

A wald.tab: a 4- or 6-column data.frame containing a pseudo-anova table for the fixed terms produced by wald.asreml.

Author(s)

Chris Brien

References

Kenward, M. G., & Roger, J. H. (1997). Small sample inference for fixed effects from restricted maximum likelihood. *Biometrics*, **53**, 983-997.

See Also

as.asrtests, testranfix.asrtests

Examples

```
## Not run:
  wald.tab <- recalcWaldTab(current.asrt,
                            dDF.fault = "supplied",
                            dDF.values = c(NA,rep(c(330,346), c(4,3))))

## End(Not run)
```

---

redoErrorIntervals.alldiffs

Adds or replaces the error intervals stored in a prediction component of an alldiffs.object.

Description

Given an alldiffs.object, adds or replaces error.intervals for its prediction component. If the backtransforms component is present, the transform.power, offset and scale will be retrieved from the backtransforms attributes, ignoring the values for the function's arguments, and the backtransformed error.intervals will also be calculated.

Usage

```
## S3 method for class 'alldiffs'
redoErrorIntervals(alldiffs.obj, error.intervals = "Confidence",
                   alpha = 0.05,
                   avsed.tolerance = 0.25, accuracy.threshold = NA,
                   LSDtype = NULL, LSDsupplied = NULL,
                   LSDby = NULL, LSDstatistic = "mean",
                   LSDaccuracy = "maxAbsDeviation",
                   retain.zeroLSDs = FALSE,
                   zero.tolerance = .Machine$double.eps ^ 0.5, ...)
```

Arguments

alldiffs.obj	An alldiffs.object.

error.intervals

A character string indicating the type of error interval, if any, to calculate in order to indicate uncertainty in the predicted values. Possible values are "none", "StandardError", "Confidence" and "halfLeastSignificant". The default is for confidence limits to be used. The "halfLeastSignificant" option results in half the Least Significant Difference (LSD) being added and subtracted to the predictions, the LSD being calculated using the square root of the mean of the variances of all or a subset of pairwise differences between the predictions. If the LSD is zero, as can happen when predictions are constrained to be equal, then the limits of the error intervals are set to NA. If LSDtype is set to overall, the avsed.tolerance is not NA and the range of the SEDs divided by the average of the SEDs exceeds avsed.tolerance then the error.intervals calculations and the plotting will revert to confidence intervals.

alpha

A numeric giving the significance level for LSDs or one minus the confidence level for confidence intervals. It is stored as an attribute to the alldiffs.object.

avsed.tolerance

A numeric giving the value of the SED range, the range of the SEDs divided by the square root of the mean of the variances of all or a subset of the pairwise differences, that is considered reasonable in calculating error.intervals. To have it ignored, set it to NA. It should be a value between 0 and 1. The following rules apply:

1. If avsed.tolerance is NA then mean LSDs of the type specified by LSDtype are calculated and used in error.intervals and plots.
2. Irrespective of the setting of LSDtype, if avsed.tolerance is not exceeded then the mean LSDs are used in error.intervals and plots.
3. If LSDtype is set to overall, avsed.tolerance is not NA, and avsed.tolerance is exceeded then error.intervals and plotting revert to confidence intervals.
4. If LSDtype is set to factor.combinations and avsed.tolerance is not exceeded for any factor combination then the half LSDs are used in error.intervals and plots; otherwise, error.intervals and plotting revert to confidence intervals.
5. If LSDtype is set to per.prediction and avsed.tolerance is not exceeded for any prediction then the half LSDs are used in error.intervals and plots; otherwise, error.intervals and plotting revert to confidence intervals.

accuracy.threshold

A numeric specifying the value of the LSD accuracy measure, which measure is specified by LSDaccuracy, as a threshold value in determining whether the hallfLeastSignificant error.interval for a predicted value is a reasonable approximation; this will be the case if the LSDs across all pairwise comparisons for which the interval's LSD was computed, as specified by LSDtype and LSDby, are similar enough to the interval's LSD, as measured by LSDaccuracy. If it is NA, it will be ignored. If it is not NA, a column of logicals named LSDwarning will be added to the predictions component of the alldiffs.object. The value of LSDwarning for a predicted.value will be TRUE if the value of the LSDaccuracy measure computed from the LSDs for differences between

this predicted.value and the other predicted.values as compared to its assignedLSD exceeds the value of accuracy.threshold. Otherwise, the value of LSDwarning for a predicted.value will be FALSE.

LSDtype
: A character string that can be overall, factor.combinations, per.prediction or supplied. It determines whether the values stored in a row of a LSD.frame are the values calculated (i) overall from the LSD values for all pairwise comparisons, unless there is only one, possibly repeated, prediction, when a notional LSD is calculated, (ii) the values calculated from the pairwise LSDs for the levels of each factor.combination, unless there is only one prediction for a level of the factor.combination, when a notional LSD is calculated, (iii) per.prediction, being based, for each prediction, on all pairwise differences involving that prediction, or (iv) as supplied values of the LSD, specified with the LSDsupplied argument; these supplied values are to be placed in the assignedLSD column of the LSD.frame stored in an alldiffs.object so that they can be used in LSD calculations.

  If LSDtype is NULL (the default), the LSDtype attribute of the alldiffs.obj will be used; it is also NULL, then the LSDtype will be set to overall.

  See LSD.frame for further information on the values in a row of this data.frame and how they are calculated.

LSDsupplied
: A data.frame or a named numeric containing a set of LSD values that correspond to the observed combinations of the values of the LSDby variables in the predictions.frame or a single LSD value that is an overall LSD. If a data.frame, it may have (i) a column for the LSDby variable and a column of LSD values or (ii) a single column of LSD values with rownames being the combinations of the observed values of the LSDby variables. Any name can be used for the column of LSD values; assignedLSD is sensible, but not obligatory. Otherwise, a numeric containing the LSD values, each of which is named for the observed combination of the values of the LSDby variables to which it corresponds. (Applying the function dae::fac.combine to the predictions component is one way of forming the required combinations for the (row) names.) The values supplied will be incorporated into assignedLSD column of the LSD.frame stored as the LSD component of the alldiffs.object.

LSDby
: A character (vector) of variables names, being the names of the factors or numerics in the classify; for each combination of their levels and values, there will be or is a row in the LSD.frame stored in the LSD component of the alldiffs.object when LSDtype is factor.combinatons.

LSDstatistic
: A character nominating one or more of minimum, q10, q25, mean, median, q75, q90 or maximum as the value(s) to be stored in the assignedLSD column in an LSD.frame; the values in the assignedLSD column are used in computing halfLeastSignificant error.intervals. Here q10, q25, q75 and q90 indicate the sample quantiles corresponding to probabilities of 0.1, 0.25, 0.75 and 0.9 for the group of LSDs from which a single LSD value is calculated. The function quantile is used to obtain them. The mean LSD is calculated as the square root of the mean of the squares of the LSDs for the group. The median is calculated using the median function. Multiple values are only produced for LSDtype set to factor.combination, in which case LSDby must not be NULL and the number of values must equal the number of observed combinations of

the values of the variables specified by LSDby. If LSDstatistic is NULL, it is reset to mean.

LSDaccuracy — A character nominating one of maxAbsDeviation, maxDeviation, q90Deviation or RootMeanSqDeviation as the statistic to be calculated as a measure of the accuracy of assignedLSD. The option q90Deviation produces the sample quantile corresponding to a probability of 0.90. The deviations are the differences between the LSDs used in calculating the LSD statistics and each assigned LSD and the accuracy is expressed as a proportion of the assigned LSD value. The calculated values are stored in the column named accuracyLSD in an LSD.frame.

retain.zeroLSDs — A logical indicating whether to retain or omit LSDs that are zero when calculating the summaries of LSDs.

zero.tolerance — A numeric specifying the value such that if an LSD is less than it, the LSD will be considered to be zero.

... — further arguments passed to recalcLSD.alldiffs.

Value

An alldiffs.object with components predictions, vcov, differences, p.differences sed, LSD and, if present in alldiffs.obj, backtransforms.

If error.intervals is not "none", then the predictions component and, if present, the backtransforms component will contain columns for the lower and upper values of the limits for the interval. The names of these columns will consist of three parts separated by full stops: 1) the first part will be lower or upper; 2) the second part will be one of Confidence, StandardError or halfLeastSignificant; 3) the third component will be limits.

The name of the response, the term, the classify and tdf, as well as the degrees of freedom of the standard error, will be set as attributes to the object. Also, if error.intervals is "halfLeastSignificant", then those of LSDtype, LSDby and LSDstatistic that are not NULL will be added as attributes of the object and of the predictions frame; additionally, LSDvalues will be added as attribute of the predictions frame, LSDvalues being the LSD values used in calculating the error.intervals.

Author(s)

Chris Brien

See Also

recalcLSD.alldiffs, exploreLSDs.alldiffs, pickLSDstatistics.alldiffs, predictPresent.asreml, plotPredictions.data.frame, allDifferences.data.frame, as.alldiffs, print.alldiffs, sort.alldiffs, subset.alldiffs, as.Date, predict.asreml

Examples

```
data(WaterRunoff.dat)

##Use asreml to get predictions and associated statistics
```

```
## Not run:
asreml.options(keep.order = TRUE) #required for asreml-R4 only
current.asr <- asreml(fixed = pH ~ Benches + (Sources * (Type + Species)),
                      random = ~ Benches:MainPlots,
                      keep.order=TRUE, data= WaterRunoff.dat)
current.asrt <- as.asrtests(current.asr, NULL, NULL)
TS.diffs <- predictPlus(classify = "Sources:Type",
                        asreml.obj = current.asr,
                        wald.tab = current.asrt$wald.tab,
                        present = c("Sources", "Type", "Species"))

## End(Not run)

## Use lmeTest and emmmeans to get predictions and associated statistics

if (requireNamespace("lmerTest", quietly = TRUE) &
    requireNamespace("emmeans", quietly = TRUE))
{
  m1.lmer <- lmerTest::lmer(pH ~ Benches + (Sources * (Type + Species)) +
                              (1|Benches:MainPlots),
                            data=na.omit(WaterRunoff.dat))
  TS.emm <- emmeans::emmeans(m1.lmer, specs = ~ Sources:Species)
  TS.preds <- summary(TS.emm)
  den.df <- min(TS.preds$df, na.rm = TRUE)
  ## Modify TS.preds to be compatible with a predictions.frame
  TS.preds <- as.predictions.frame(TS.preds, predictions = "emmean",
                                   se = "SE", interval.type = "CI",
                                   interval.names = c("lower.CL", "upper.CL"))

  ## Form an all.diffs object and check its validity
  TS.vcov <- vcov(TS.emm)
  TS.diffs <- allDifferences(predictions = TS.preds,
                             classify = "Sources:Species",
                             vcov = TS.vcov, tdf = den.df)
  validAlldiffs(TS.diffs)
}

## Plot p-values for predictions obtained using asreml or lmerTest
if (exists("TS.diffs"))
{
  ##Recalculate the LSD values for predictions obtained using asreml or lmerTest
  TS.diffs <- redoErrorIntervals.alldiffs(TS.diffs,
                                          error.intervals = "halfLeastSignificant")

}
```

REMLRT.asreml *Performs a REML ratio test to compare two models.*

Description

Extracts the REML log likelihood and the number of variance parameters from two asreml objects. It assumes that the first asreml object corresponds to the null hypothesis and the second asreml object to the alternative hypothesis for the test being conducted. That is, the second asreml object is the result of fitting a model that is a reduced version of the model for the first object. In the case where the reduced model is obtained by setting positively-constrained variance parameters in the full model to zero, the positive.zero argument should be set to TRUE so that the p-value is computed using a mixture of chi-square distributions as described in Self and Liang (1987).

The function checks that the models do not differ in either their fixed or sparse models.

Usage

```
## S3 method for class 'asreml'
REMLRT(h0.asreml.obj, h1.asreml.obj,
       positive.zero = FALSE, bound.test.parameters = "none",
       DF = NULL, bound.exclusions = c("F","B","S","C"), ...)
```

Arguments

h0.asreml.obj	asreml object containing the fit under the model for the null hypothesis.
h1.asreml.obj	asreml object containing the fit under the model for the alternative hypothesis.
positive.zero	Indicates whether the hypothesized values for the variance components being tested are on the boundary of the parameter space. For example, this is true for positively-constrained variance components that, under the reduced model, are zero. This argument does not need to be set if bound.test.parameters is set.
bound.test.parameters	Indicates whether for the variance components being tested, at least some of the hypothesized values are on the boundary of the parameter space. The possibilities are "none", "onlybound" and "one-and-one". The default is "none", although if it is set to "none" and positive.zero is TRUE then bound.test.parameters is taken to be "onlybound". When bound.test.parameters is set to "one-and-one", it signifies that there are two parameters being tested, one of which is bound and the other is not. For example, the latter is true for testing a covariance and a positively-constrained variance component that, under the reduced model, are zero.
DF	A numeric giving the difference between the two models in the number of variance parameters whose estimates are not of the type specified in bound.exclusions. If NULL then this is determined from the information in full.asreml.obj and reduced.asreml.obj.
bound.exclusions	A character specifying one or more bound (constraint) codes that will result in a variance parameter being excluded from the count of estimated variance parameters. If set to NULL then none will be excluded.
...	Provision for passing arguments to functions called internally - not used at present.

Value

A `data.frame` containing the log of the likelihood ratio, its degrees of freedom, its p-value and the number of bound parameters in each of the two models being compared.

Note

If `DF` is not `NULL`, the supplied value is used. Otherwise `DF` is determined from the information in `h1.asreml.obj` and `h0.asreml.obj`. In this case, the degrees of freedom for the test are computed as the difference between the two models in the number of variance parameters whose estimates do not have a code for bound specified in `bound.exclusions`.

If ASReml-R version 4 is being used then the codes specified in `bound.exclusions` are not restricted to a subset of the default codes, but a warning is issued if a code other than these is specified. For ASReml-R version 3, only a subset of the default codes are allowed: `F` (`Fixed`), `B` (`Boundary`), `C` (`Constrained`) and `S` (`Singular`).

The test statistic is calculated as $2(log(REML)_1 - log(REML)_0)$.

This procedure is only appropriate when the null hypothesis is that (i) all parameters are on the boundary of the parameter space (ii) all parameters are in the interior of the parameter space, or (iii) there are two parameters, one of which is on the boundary and the other is not. Other cases have been discussed by Self and Liang (1987), but are not implemented here.

Author(s)

Chris Brien

References

Self, S.G., and Liang, K-Y. (1987) Asymptotic Properties of Maximum Likelihood Estimators and Likelihood Ratio Tests Under Nonstandard Conditions. *Journal of the American Statistical Association*, **82**, 605-10.

See Also

`infoCriteria.asreml`, `testranfix.asrtests`

Examples

```
## Not run:
    REMLRT(ICV.max, ICV.red, bound.test.parameters = "onlybound")

## End(Not run)
```

renewClassify.alldiffs

Renews the components in an alldiffs.object *according to a new classify.*

Description

The classify is an attribute of an alldiffs.object and determines the order within the components of an unsorted alldiffs.object. This function resets the classify attribute and re-orders the components of alldiffs.object to be in standard order for the variables in a newclassify, using allDifferences.data.frame. The newclassify may be just a re-ordering of the variable names in the previous classify, or be based on a new set of variable names. The latter is particularly useful when linTransform.alldiffs has been used with a matrix and it is desired to replace the resulting Combination classify with a newclassify comprised of a more meaningful set of variables; first replace Combination in the predictions component with the new set of variables and then call renewClassify.

Usage

```
## S3 method for class 'alldiffs'
renewClassify(alldiffs.obj, newclassify,
              sortFactor = NULL, sortParallelToCombo = NULL,
            sortNestingFactor = NULL, sortOrder = NULL, decreasing = FALSE, ...)
```

Arguments

alldiffs.obj	An alldiffs.object.
newclassify	A character string giving the variables that define the margins of the multiway table that was predicted, but ordered so that the predictions are in the desired order when they are arranged in standard order for the newclassify. Multiway tables are specified by forming an interaction type term from the classifying variables, that is, separating the variable names with the : operator. The number of combined values of the set of variable name(s) must equal the number of rows in the predictions component.
sortFactor	A character containing the name of the factor that indexes the set of predicted values that determines the sorting of the components. If there is only one variable in the classify term then sortFactor can be NULL and the order is defined by the complete set of predicted values. If there is more than one variable in the classify term then sortFactor must be set. In this case the sortFactor is sorted in the same order within each combination of the values of the sortParallelToCombo variables: the classify variables, excluding the sortFactor. There should be only one predicted value for each unique value of sortFactor within each set defined by a combination of the values of the classify variables, excluding the sortFactor factor. The order to use is determined by either sortParallelToCombo or sortOrder.

sortParallelToCombo
: A list that specifies a combination of the values of the factors and numerics, excluding sortFactor, that are in classify. Each of the components of the supplied list is named for a classify variable and specifies a single value for it. The combination of this set of values will be used to define a subset of the predicted values whose order will define the order of sortFactor. Each of the other combinations of the values of the factors and numerics will be sorted in parallel. If sortParallelToCombo is NULL then the first value of each classify variable, except for the sortFactor factor, in the predictions component is used to define sortParallelToCombo. If there is only one variable in the classify then sortParallelToCombo is ignored.

sortNestingFactor
: A character containing the name of the factor that defines groups of the sortFactor within which the predicted values are to be ordered. If there is only one variable in the classify then sortNestingFactor is ignored.

sortOrder
: A character vector whose length is the same as the number of levels for sortFactor in the predictions component of the alldiffs.object. It specifies the desired order of the levels in the reordered components of the alldiffs.object. The argument sortParallelToCombo is ignored.

  The following creates a sortOrder vector levs for factor f based on the values in x: levs <- levels(f)[order(x)].

decreasing
: A logical passed to order that detemines whether the order is for increasing or decreasing magnitude of the predicted values.

...
: further arguments passed to allDifferences.data.frame; attributes tranform.power, offset and scale cannot be passed.

Details

First, the components of the alldiffs.object is arranged in standard order for the newclassify. Then predictions are reordered according to the settings of sortFactor, sortParallelToCombo, sortOrder and decreasing (see sort.alldiffs for details).

Value

The alldiffs.object supplied with the following components, if present, sorted: predictions, vcov, backtransforms, differences, p.differences and sed. Also, the sortFactor and sortOrder attributes are set.

Author(s)

Chris Brien

See Also

as.alldiffs, allDifferences.data.frame, print.alldiffs, sort.alldiffs, redoErrorIntervals.alldiffs, recalcLSD.alldiffs, predictPlus.asreml, predictPresent.asreml

Examples

```
data(WaterRunoff.dat)

##Use asreml to get predictions and associated statistics

## Not run:
#Analyse pH
m1.asr <- asreml(fixed = pH ~ Benches + (Sources * (Type + Species)),
                 random = ~ Benches:MainPlots,
                 keep.order=TRUE, data= WaterRunoff.dat)
current.asrt <- as.asrtests(m1.asr, NULL, NULL)
current.asrt <- as.asrtests(m1.asr)
current.asrt <- rmboundary(current.asrt)
m1.asr <- current.asrt$asreml.obj

#Get predictions and associated statistics
TS.diffs <- predictPlus.asreml(classify = "Sources:Type",
                               asreml.obj = m1.asr, tables = "none",
                               wald.tab = current.asrt$wald.tab,
                               present = c("Type","Species","Sources"))

## End(Not run)

## Use lmeTest and emmmeans to get predictions and associated statistics

if (requireNamespace("lmerTest", quietly = TRUE) &
    requireNamespace("emmeans", quietly = TRUE))
{
  #Analyse pH
  m1.lmer <- lmerTest::lmer(pH ~ Benches + (Sources * (Type + Species)) +
                              (1|Benches:MainPlots),
                            data=na.omit(WaterRunoff.dat))
  TS.emm <- emmeans::emmeans(m1.lmer, specs = ~ Sources:Type)
  TS.preds <- summary(TS.emm)
  den.df <- min(TS.preds$df, na.rm = TRUE)
  ## Modify TS.preds to be compatible with a predictions.frame
  TS.preds <- as.predictions.frame(TS.preds, predictions = "emmean",
                                   se = "SE", interval.type = "CI",
                                   interval.names = c("lower.CL", "upper.CL"))

  ## Form an all.diffs object and check its validity
  TS.vcov <- vcov(TS.emm)
  TS.diffs <- allDifferences(predictions = TS.preds,
                               classify = "Sources:Type",
                               vcov = TS.vcov, tdf = den.df)
  validAlldiffs(TS.diffs)
}

#Re-order predictions from asreml or lmerTest so all Sources for the same Type are together
#for each combination of A and B
if (exists("TS.diffs"))
{
```

```
  TS.diffs.reord <- renewClassify(TS.diffs, newclassify = "Type:Sources")
  validAlldiffs(TS.diffs.reord)
}
```

`reparamSigDevn.asrtests`

Reparamterizes each random (deviations) term involving `devn.fac` *to a fixed term and ensures that the same term, with* `trend.num` *replacing* `devn.fac`*, is included if any other term with* `trend.num` *is included in* `terms`.

Description

This function reparamterizes each random (deviations) term involving `devn.fac` to a fixed term and ensures that the same term with `trend.num` replacing `devn.fac` is included if any other term with `trend.num` is included in `terms`. It also ansures that any term with `spl{trend.num}` replacing `devn.fac` in a term being reparameterized is removed from the model.

Usage

```
## S3 method for class 'asrtests'
reparamSigDevn(asrtests.obj,terms = NULL,
               trend.num = NULL, devn.fac = NULL,
               allow.unconverged = TRUE, allow.fixedcorrelation = TRUE,
               checkboundaryonly = FALSE,
               denDF = "numeric", IClikelihood = "none",
               trace = FALSE, update = TRUE,
               set.terms = NULL, ignore.suffices = TRUE,
               bounds = "P", initial.values = NA,...)
```

Arguments

`asrtests.obj`	an `asrtests.object` containing the components (i) `asreml.obj`, (ii) `wald.tab`, and (iii) `test.summary`.
`terms`	A character string vector giving the random terms that are to be reparameterized.
`trend.num`	A character string giving the name of the numeric covariate that corresponds to `devn.fac` and is potentially included in terms in the fitted model.
`devn.fac`	A character string giving the name of the factor that corresponds to `trend.num` and is included in terms in the fitted model. The name must match those in the `vparameters` component of the `asreml.obj` component in the `asrtests.obj`.
`allow.unconverged`	A `logical` indicating whether to accept a new model even when it does not converge. Initially all changes are made with `allow.unconverged` set to `TRUE`. If `allow.unconverged` has been set to `FALSE` in the call and the final fit does not converge, an attempt is made to achieve convergence by removing any boundary terms. If this is unsuccessful, the supplied `asrtests.obj` is returned.

allow.fixedcorrelation
: A logical indicating whether to accept a new model even when it contains correlations in the model whose values have been designated as fixed, bound or singular. If FALSE and the new model contains correlations whose values have not been able to be estimated, the supplied asrtests.obj is returned. The fit in the asreml.obj component of the supplied asrtests.obj will also be tested and a warning issued if both fixed correlations are found in it and allow.fixedcorrelation is FALSE.

checkboundaryonly
: If TRUE then boundary and singular terms are not removed by rmboundary.asrtests; a warning is issued instead.

denDF
: Specifies the method to use in computing approximate denominator degrees of freedom when wald.asreml is called. Can be none to suppress the computations, numeric for numerical methods, algebraic for algebraic methods or default, the default, to autommatically choose numeric or algebraic computations depending on problem size. The denominator degrees of freedom are calculated according to Kenward and Roger (1997) for fixed terms in the dense part of the model.

IClikelihood
: A character that controls both the occurrence and the type of likelihood for information criterion in the test.summary of the new asrtests.object. If none, none are included. Otherwise, if REML, then the AIC and BIC based on the Restricted Maximum Likelihood are included; if full, then the AIC and BIC based on the full likelihood, evaluated using REML estimates, are included. (See also infoCriteria.asreml.)

trace
: If TRUE then partial iteration details are displayed when ASReml-R functions are invoked; if FALSE then no output is displayed.

update
: If TRUE, and set.terms is NULL, then newfit.asreml is called to fit the model to be tested, using the values of the variance parameters stored in the asreml.object, that is stored in asrtests.obj, as starting values. If FALSE or set.terms is not NULL, then newfit.asreml will not use the stored variance parameter values as starting values when fitting the new model, the only modifications being (i) the models are updated and (ii) those specified via ....

set.terms
: A character vector specifying the terms that are to have bounds and/or initial values set prior to fitting.

ignore.suffices
: A logical vector specifying whether the suffices of the asreml-assigned names of the variance terms (i.e. the information to the right of an "!", other than "R!") is to be ignored in matching elements of terms. If TRUE for an element of terms, the suffices are stripped from the asreml-assigned names. If FALSE for an element of terms, the element must exactly match an asreml-assigned name for a variance term. This vector must be of length one or the same length as terms. If it is of length one then the same action is applied to the asreml-assigned suffices for all the terms in terms.

bounds
: A character vector specifying the bounds to be applied to the terms specified in set.terms. This vector must be of length one or the same length as set.terms. If it is of length one then the same constraint is applied to all the

terms in set.terms. If any of the bounds are equal to NA then they are left unchanged for those terms.

initial.values A character vector specifying the initial values for the terms specified in terms. This vector must be of length one or the same length as terms. If it is of length one then the same initial value is applied to all the terms in terms. If any of the initial.values are equal to NA then they are left unchanged for those terms.

... further arguments passed to asreml via changeTerms.asrtests and as.asrtests.

Value

An asrtests.object containing the components (i) asreml.obj, (ii) wald.tab, and (iii) test.summary.

Author(s)

Chris Brien

References

Kenward, M. G., & Roger, J. H. (1997). Small sample inference for fixed effects from restricted maximum likelihood. *Biometrics*, **53**, 983-997.

See Also

as.asrtests, changeTerms.asrtests, testranfix.asrtests, testresidual.asrtests, newfit.asreml, chooseModel.asrtests

Examples

```
## Not run:
  data(WaterRunoff.dat)
  asreml.options(keep.order = TRUE) #required for asreml-R4 only
  current.asr <- asreml(fixed = log.Turbidity ~ Benches + Sources + Type + Species +
                            Sources:Type + Sources:Species + Sources:Species:xDay +
                            Sources:Species:Date,
                        data = WaterRunoff.dat, keep.order = TRUE)
  current.asrt <- as.asrtests(current.asr, NULL, NULL)

  #Examine terms that describe just the interactions of Date and the treatment factors
  terms.treat <- c("Sources", "Type", "Species", "Sources:Type", "Sources:Species")
  date.terms <- sapply(terms.treat,
                       FUN=function(term){paste("Date:",term,sep="")},
                       simplify=TRUE)
  date.terms <- c("Date", date.terms)
  date.terms <- unname(date.terms)
  treat.marginality <-  matrix(c(1,0,0,0,0,0, 1,1,0,0,0,0,  1,0,1,0,0,0,
                                 1,0,1,1,0,0, 1,1,1,0,1,0, 1,1,1,1,1,1), nrow=6)
  rownames(treat.marginality) <- date.terms
  colnames(treat.marginality) <- date.terms
  choose <- chooseModel(current.asrt, treat.marginality, denDF="algebraic")
  current.asrt <- choose$asrtests.obj
  current.asr <- current.asrt$asreml.obj
```

```
  sig.date.terms <- choose$sig.terms

  #Remove all Date terms left in the fixed model
  terms <- "(Date/(Sources * (Type + Species)))"
  current.asrt <- changeTerms(current.asrt, dropFixed = terms)
  #if there are significant date terms, reparameterize to xDays + spl(xDays) + Date
  if (length(sig.date.terms) != 0)
  { #add lin + spl + devn for each to fixed and random models
    trend.date.terms <- sapply(sig.date.terms,
                               FUN=function(term){sub("Date","xDay",term)},
                               simplify=TRUE)
    trend.date.terms <- paste(trend.date.terms,  collapse=" + ")
    current.asrt <- changeTerms(current.asrt, addFixed=trend.date.terms)
    trend.date.terms <- sapply(sig.date.terms,
                               FUN=function(term){sub("Date","spl(xDay)",term)},
                               simplify=TRUE)
    trend.date.terms <- c(trend.date.terms, sig.date.terms)
    trend.date.terms <- paste(trend.date.terms,  collapse=" + ")
    current.asrt <- changeTerms(current.asrt, addRandom = trend.date.terms)
    current.asrt <- rmboundary(current.asrt)
  }

  #Now test terms for sig date terms
  spl.terms <- sapply(terms.treat,
                      FUN=function(term){paste("spl(xDay):",term,sep="")},
                      simplify=TRUE)
  spl.terms <- c("spl(xDay)",spl.terms)
  lin.terms <- sapply(terms.treat,
                      FUN=function(term){paste(term,":xDay",sep="")},
                      simplify=TRUE)
  lin.terms <- c("xDay",lin.terms)
  systematic.terms <- c(terms.treat, lin.terms, spl.terms, date.terms)
  systematic.terms <- unname(systematic.terms)
  treat.marginality <-  matrix(c(1,0,0,0,0,0, 1,1,0,0,0,0,  1,0,1,0,0,0,
                                 1,0,1,1,0,0, 1,1,1,1,1,0, 1,1,1,1,1,1), nrow=6)
  systematic.marginality <- kronecker(matrix(c(1,0,0,0, 1,1,0,0,
                                               1,1,1,0, 1,1,1,1), nrow=4),
                                      treat.marginality)
  systematic.marginality <- systematic.marginality[-1, -1]
  rownames(systematic.marginality) <- systematic.terms
  colnames(systematic.marginality) <- systematic.terms
  choose <- chooseModel(current.asrt, systematic.marginality,
                        denDF="algebraic", pos=TRUE)
  current.asrt <- choose$asrtests.obj

  #Check if any deviations are significant and, for those that are, go back to
  #fixed dates
  current.asrt <- reparamSigDevn(current.asrt, choose$sig.terms,
                                 trend.num = "xDay", devn.fac = "Date",
                                 denDF = "algebraic")

## End(Not run)
```

rmboundary.asrtests *Removes any boundary or singular variance components from the fit stored in* asrtests.obj *and records their removal in an* asrtests.object.

Description

Any terms specified in the random model that are estimated on the boundary or are singular and can be removed are removed from the fit stored in the asreml object stored in the asrtests.object. Terms that specify multiple parameters in the random model cannot be removed (e.g. terms specified using the at function with more than one level of the factor) and terms in residual model are not removed. Terms that can be removed are selected for removal in the following order based on whether they involve: (i) a dev function, (ii) only factors, (iii) an spl function, (iv) a pol function and (v) a lin function or a variable that is an integer or a numeric. It should be noted that this order of removal presumes that random deviation terms are specified via the dev function rather than via a random factor. Once the earliest of the above classes with a boundary term is identified, a term within this class is selected for removal. For all classes, except for factor-only terms, the smallest term with the largest number of variables/factors is removed. Amongst factor-only terms, the smallest term with the smallest number of variables/factors is removed. After each variance component is removed, a row for it is added to the test.summary data.frame and the model refitted. If there are further boundary or singular terms, one is removed using the above strategy. This process continues until there are no further boundary or singular variance components that are removable. Other types of boundary or singular terms, which cannot be removed, are reported in warning messages.

Usage

```
## S3 method for class 'asrtests'
rmboundary(asrtests.obj, checkboundaryonly = FALSE,
           IClikelihood = "none", trace = FALSE, update = TRUE,
           set.terms = NULL, ignore.suffices = TRUE,
           bounds = "P", initial.values = NA, ...)
```

Arguments

asrtests.obj	an asrtests.object containing the components (i) asreml.obj, (ii) wald.tab, and (iii) test.summary.
checkboundaryonly	If TRUE then boundary and singular terms are not removed by rmboundary.asrtests; a warning is issued instead.
IClikelihood	A character that controls both the occurrence and the type of likelihood for information criterion in the test.summary of the new asrtests.object. If none, none are included. Otherwise, if REML, then the AIC and BIC based on the Restricted Maximum Likelihood are included; if full, then the AIC and BIC based on the full likelihood, evaluated using REML estimates, are included. (See also infoCriteria.asreml.)

trace	If TRUE then partial iteration details are displayed when ASReml-R functions are invoked; if FALSE then no output is displayed.
update	If TRUE, and set.terms is NULL, then newfit.asreml is called to fit the model with any boundary terms removed and using the values of the variance parameters stored in the asreml.object, that is stored in asrtests.obj, as starting values. If FALSE or set.terms is not NULL, then newfit.asreml will not use the stored variance parameter values as starting values when fitting the model without the boundary terms.
set.terms	A character vector specifying the terms that are to have bounds and/or initial values set prior to fitting. The names must match those in the vparameters component of the asreml.obj component in the asrtests.object.
ignore.suffices	A logical vector specifying whether the suffices of the asreml-assigned names of the variance terms (i.e. the information to the right of an "!", other than "R!") is to be ignored in matching elements of terms. If TRUE for an element of terms, the suffices are stripped from the asreml-assigned names. If FALSE for an element of terms, the element must exactly match an asreml-assigned name for a variance term. This vector must be of length one or the same length as terms. If it is of length one then the same action is applied to the asreml-assigned suffices for all the terms in terms.
bounds	A character vector specifying the bounds to be applied to the terms specified in set.terms. This vector must be of length one or the same length as set.terms. If it is of length one then the same constraint is applied to all the terms in set.terms. If any of the bounds are equal to NA then they are left unchanged for those terms.
initial.values	A character vector specifying the initial values for the terms specified in terms. This vector must be of length one or the same length as terms. If it is of length one then the same initial value is applied to all the terms in terms. If any of the initial.values are equal to NA then they are left unchanged for those terms.
...	Further arguments, including asreml arguments, passed to newfit.asreml.

Value

An asrtests.object containing the components (i) asreml.obj, (ii) wald.tab, and (iii) test.summary.

Author(s)

Chris Brien

See Also

as.asrtests, changeTerms.asrtests, testranfix.asrtests, testresidual.asrtests, newfit.asreml, reparamSigDevn.asrtests, chooseModel.asrtests

Examples

```
## Not run:
current.asrt <- rmboundary(current.asrt)
```

```
## End(Not run)
```

`setvarianceterms.call` *allows the setting of bounds and initial values for terms in the* `random` *and* `residual` *arguments of an* `asreml` *call, with the resulting call being evaluated.*

Description

Takes an unevaluated `call` and evaluates the `call` after setting the bounds and initial values for the terms specified in `terms`. The elements of `terms` are matched with those generated by asreml and used, for example, in the `varcomp` component of a `summary.asreml` object. These names generally include descriptive suffices. To match an element of `terms` that includes such a suffix, set `ignore.suffices` to `FALSE` so that a literal match between the element and the assigned names is sought.

Note that the `terms`, `bounds` and `initial.values` are stored on entry in a `data.frame`, named `setvparameters`, as a component in `call` that is itself a component of the `asreml.obj` that is returned. The `data.frame` `setvparameters` contains all of the values of `terms`, `ignore.suffices`, `bounds` and `initial.values` that have been set in this and previous calls to `setvarianceterms.call` and other model modification and selection functions in `asremlPlus`, for example changeModelOnIC.asrtests, testranfix.asrtests and changeTerms.asrtests. It is used in subsequent calls to model modification and selection functions to ensure that the bounds and initial values that have been set are retained in new model fits.

Usage

```
## S3 method for class 'call'
setvarianceterms(call, terms, ignore.suffices = TRUE,
                 bounds = "P", initial.values = NA, ...)
```

Arguments

`call`	an unevaluated `call` to `asreml`. One way to create such a `call` is to use the `call` function with its `name` argument set to `"asreml"`. Another is to obtain it from the `call` component of an `asreml` object (e.g. `call <- asreml.obj$call`).
`terms`	A character vector specifying the terms that are to have bounds and/or initial values specified. The names must match those in the `vparameters` component of the `asreml.obj` component in the `asrtests.object`.
`ignore.suffices`	A logical vector specifying whether the suffices of the `asreml`-assigned names of the variance terms (i.e. the information to the right of an "!", other than "R!") is to be ignored in matching elements of `terms`. If `TRUE` for an element of `terms`, the suffices are stripped from the `asreml`-assigned names. If `FALSE` for an element of `terms`, the element must exactly match an `asreml`-assigned name for a variance term. This vector must be of length one or the same length as `terms`. If it is of length one then the same action is applied to the `asreml`-assigned suffices for all the terms in `terms`.

bounds — A character vector specifying the bounds to be applied to the terms specified in terms. This vector must be of length one or the same length as terms. If it is of length one then the same constraint is applied to all the terms in terms. If any of the bounds are equal to NA then they are left unchanged for those terms.

The codes used by ASReml are:

- B - fixed at a boundary;
- F - fixed by the user;
- P - positive definite;
- C - Constrained by user;
- U - unbounded.

initial.values — A character vector specifying the initial values for the terms specified in terms. This vector must be of length one or the same length as terms. If it is of length one then the same initial value is applied to all the terms in terms. If any of the initial.values are equal to NA then they are left unchanged for those terms.

... — additional arguments to be added to the asreml call, or arguments in the asreml call with changed values.

Value

An asreml object, with the updated setvparameters data.frame, stored in the call component of the asreml object.

Author(s)

Chris Brien

References

Butler, D. G., Cullis, B. R., Gilmour, A. R., Gogel, B. J. and Thompson, R. (2023). *ASReml-R Reference Manual Version 4.2.* VSN International Ltd, https://asreml.kb.vsni.co.uk/.

See Also

newfit.asreml, update.asreml

Examples

```
## Not run:
   m1.call <- call("asreml",
                   fixed = Height ~ (Block + Irrig)*csDay.num,
                   random= ~ spl(csDay.num)/(Irrig+Block)
                             + dev(csDay.num)
                             + str(~Block:Plot/csDay.num, ~us(2):id(20))
                             + Block:Plot:spl(csDay.num),
                   data=quote(dat)) ##use quote to stop evaluation of dat here
   terms <- c("Block:Plot+Block:Plot:csDay.num!us(2).2:1", "R!variance")
   m1.asreml <- setvarianceterms(m1.call, terms, bounds=c("U","P"),
                                 initial=c(NA,3), ignore.suffices=c(FALSE,TRUE))
   summary(m1.asreml)
```

```
## End(Not run)
```

`simulate.asreml` *Produce sets of simulated data from a multivariate normal distribution and save quantities related to the simulated data*

Description

Produce in parallel sets of simulated data corresponding to an `asreml` model, along with its fitted values and residuals. A variance matrix `V`, corresponding to the `random` and `residual` models must be supplied. What to save is specified by the `which` argument.

Usage

```
## S3 method for class 'asreml'
simulate(object, nsim=100, seed = NULL, means=NULL, V, tolerance = 1E-10,
         update = TRUE, trace = FALSE, which="data", units = "ignore",
         ncores = 2, ...)
```

Arguments

`object`	An `asreml` object from a call to `asreml` in which the `data` argument has been set.
`means`	The `vector` of means to be used in generating simulated data sets. If it is `NULL`, the fitted values based on `object` are used. It must be the same length as the response variable for `object`.
`V`	The fitted variance `matrix`, i.e. having the pattern and values that conform to the model fit stored in the supplied `object`.
`nsim`	The number of data sets to be simulated.
`seed`	A single value, interpreted as an integer, that specifies the starting value of the random number generator. The "L'Ecuyer-CMRG" random generator is used and `nextRNGStream` is used to seed each core from the original `seed`.
`tolerance`	The value such that eigenvalues less than it are considered to be zero.
`update`	If `TRUE` then the arguments `R.param` and `G.param` are set to those in the `asreml` object supplied in `object` so that the values from the original model are used as starting values. If `FALSE` then `asreml` `calls` are evaluated, the only changes from the previous call being that (i) the model is fitted to simulated data and (ii) modifications specified via `...` are mode, except that changes cannot be made to any of the models.
`trace`	If `TRUE` then partial iteration details are displayed when ASReml-R functions are invoked; if `FALSE` then no output is displayed.

which	The quantities from the simulated data set to be stored. Any combination of "response", "residuals" and "fitted", or "all". If residuals and/or fitted is specified, those for the analysis stored in object will be added to the data.frame nominated in the data argument of object and the modified data.frame added as a component named data in the list that is the value returned by the function.
units	A character indicating whether the BLUPs for units are added to the residuals when this reserved factor is included in the random model. Possible values are addtoresiduals and ignore.
ncores	A numeric specifying the number of cores to use in doing the simulations. In choosing a value for ncores, it is necessary to take into account other processes that are using parallel processing at the same time.
...	Other arguments that are passed down to the function asreml. Changes to the models are not allowed. Other changes are dangerous and generally should be avoided.

Details

Generate nsim sets of data and analyse them using asreml using the model in object, performing the generation and analysis of several sets in parallel. Note, if the analysis for a data set does not converge in maxiter iterations, it is discarded and a replacement data set generated. The value of maxiter can be specified in the call to simulate.asreml. The fitted values and residuals are extracted as required. If aom = TRUE when the simulated data are analysed, standardised conditional residuals are stored. If which includes residuals or fitted, the specified quantities for the observed data are added to the data.frame on which the fit in object is based.

Value

A list with the following components whose presence depends on the setting of which:

1. **observed:** present if which includes residuals or fitted, in which case it will be the data.frame on which the fit in object is based, with residuals and/or fitted.
2. **data:** present if which includes data, a data.frame containing the simulated data sets.
3. **fitted:** present if which includes fitted, a data.frame containing the fitted values from the analyses of the simulated data sets.
4. **residuals:** present if which includes residuals, a data.frame containing the residuals from the analyses of the simulated data sets.

Author(s)

Chris Brien

See Also

asreml, newfit.asreml, variofaces.asreml, plotVariofaces.data.frame, set.seed.

Examples

```
## Not run:
data(Wheat.dat)
current.asr <- asreml(yield ~ Rep + WithinColPairs + Variety,
                      random = ~ Row + Column + units,
                      residual = ~ ar1(Row):ar1(Column),
                      data=Wheat.dat)
current.asrt <- as.asrtests(current.asr, NULL, NULL)
current.asrt <- rmboundary.asrtests(current.asrt)
# Form variance matrix based on estimated variance parameters
s2 <- current.asr$sigma2
gamma.Row <- current.asr$gammas[1]
gamma.unit <- current.asr$gammas[2]
rho.r <- current.asr$gammas[4]
rho.c <- current.asr$gammas[5]
row.ar1 <- mat.ar1(order=10, rho=rho.r)
col.ar1 <- mat.ar1(order=15, rho=rho.c)
V <- gamma.Row * fac.sumop(Wheat.dat$Row) +
     gamma.unit * diag(1, nrow=150, ncol=150) +
     mat.dirprod(col.ar1, row.ar1)
V <- s2*V

#Produce residuals from 100 simulated data sets
resid <- simulate(current.asr, V=V, which="residuals", ncores = parallel::detectCores())

## End(Not run)
```

`sort.alldiffs` *Sorts the components in an* `alldiffs.object` *according to the predicted values associated with a factor.*

Description

Sorts the rows of the components in an `alldiffs.object` (see `as.alldiffs`) that are `data.frames` and the rows and columns of those that are `matrices` according to the predicted values in the `predictions` component. These predicted values are generally obtained using `predict.asreml` by specifying a `classify` term comprised of one or more variables. Generally, the values associated with one variable are sorted in parallel within each combination of values of the other variables. When there is more than one variable in the `classify` term, the sorting is controlled using one or more of `sortFactor`, `sortParallelToCombo` and `sortOrder`. If there is only one variable in the `classify` then all components are sorted according to the order of the complete set of predictions.

Note that `renewClassify.alldiffs` is called after sorting to ensure that the order of the rows and columns of the components is in standard order for the new variable order.

Usage

```
## S3 method for class 'alldiffs'
sort(x, decreasing = FALSE, classify = NULL, sortFactor = NULL,
```

```
sortParallelToCombo = NULL, sortNestingFactor = NULL,
sortOrder = NULL, ...)
```

Arguments

x	An alldiffs.object.
decreasing	A logical passed to order that detemines whether the order is for increasing or decreasing magnitude of the predicted values.
classify	A character string giving the variables that define the margins of the multiway table that was predicted. Multiway tables are specified by forming an interaction type term from the classifying variables, that is, separating the variable names with the : operator. If NULL, it will be obtained from the classify attribute of the as.alldiffs object supplied through x.
sortFactor	A character containing the name of the factor that indexes the set of predicted values that determines the sorting of the components. If there is only one variable in the classify term then sortFactor can be NULL and the order is defined by the complete set of predicted values. If there is more than one variable in the classify term then sortFactor must be set. In this case the sortFactor is sorted in the same order within each combination of the values of the sortParallelToCombo variables: the classify variables, excluding the sortFactor. There should be only one predicted value for each unique value of sortFactor within each set defined by a combination of the values of the classify variables, excluding the sortFactor factor. The order to use is determined by either sortParallelToCombo or sortOrder.
sortParallelToCombo	A list that specifies a combination of the values of the factors and numerics, excluding sortFactor, that are in classify. Each of the components of the supplied list is named for a classify variable and specifies a single value for it. The combination of this set of values will be used to define a subset of the predicted values whose order will define the order of sortFactor. Each of the other combinations of the values of the factors and numerics will be sorted in parallel. If sortParallelToCombo is NULL then the first value of each classify variable, except for the sortFactor factor, in the predictions component is used to define sortParallelToCombo. If there is only one variable in the classify then sortParallelToCombo is ignored.
sortNestingFactor	A character containing the name of the factor that defines groups of the sortFactor within which the predicted values are to be ordered. If there is only one variable in the classify then sortNestingFactor is ignored.
sortOrder	A character vector whose length is the same as the number of levels for sortFactor in the predictions component of the alldiffs.object. It specifies the desired order of the levels in the reordered components of the alldiffs.object. The argument sortParallelToCombo is ignored. The following creates a sortOrder vector levs for factor f based on the values in x: levs <- levels(f)[order(x)].
...	further arguments passed to or from other methods. Not used at present.

Details

The basic technique is to change the order of the levels of the sortFactor within the predictions and, if present, backtransforms components so that they are ordered for a subset of predicted values, one for each levels of the sortFactor. When the classify term consists of more than one variable then a subset of one combination of the values of variables other than the sortFactor, the sortParallelToCombo combination, must be chosen for determining the order of the sortFactor levels. Then the sorting of the rows (and columns) will be in parallel within each combination of the values of sortParallelToCombo variables: the classify term, excluding the sortFactor.

Value

The alldiffs.object supplied with the following components, if present, sorted: predictions, vcov, backtransforms, differences, p.differences and sed. Also, the sortFactor and sortOrder attributes are set.

Author(s)

Chris Brien

See Also

as.alldiffs, allDifferences.data.frame, print.alldiffs, sort.predictions.frame, renewClassify.alldiffs, redoErrorIntervals.alldiffs, recalcLSD.alldiffs, predictPlus.asreml, predictPresent.asreml

Examples

```
##Halve WaterRunoff data to reduce time to execute
data(WaterRunoff.dat)
tmp <- subset(WaterRunoff.dat, Date == "05-18")

##Use asreml to get predictions and associated statistics

## Not run:
#Analyse pH
m1.asr <- asreml(fixed = pH ~ Benches + (Sources * (Type + Species)),
                 random = ~ Benches:MainPlots,
                 keep.order=TRUE, data= tmp)
current.asrt <- as.asrtests(m1.asr, NULL, NULL)
current.asrt <- as.asrtests(m1.asr)
current.asrt <- rmboundary(current.asrt)
m1.asr <- current.asrt$asreml.obj

#Get predictions and associated statistics
TS.diffs <- predictPlus.asreml(classify = "Sources:Type",
                               asreml.obj = m1.asr, tables = "none",
                               wald.tab = current.asrt$wald.tab,
                               present = c("Type","Species","Sources"))

#Use sort.alldiffs and save order for use with other response variables
TS.diffs.sort <- sort(TS.diffs, sortFactor = "Sources",
```

```
                      sortParallelToCombo = list(Type = "Control"))
sort.order <- attr(TS.diffs.sort, which = "sortOrder")

#Analyse Turbidity
m2.asr <- asreml(fixed = Turbidity ~ Benches + (Sources * (Type + Species)),
                 random = ~ Benches:MainPlots,
                 keep.order=TRUE, data= tmp)
current.asrt <- as.asrtests(m2.asr)
#Use pH sort.order to sort Turbidity alldiffs object
diffs2.sort <- predictPlus(m2.asr, classify = "Sources:Type",
                           pairwise = FALSE, error.intervals = "Stand",
                           tables = "none", present = c("Type","Species","Sources"),
                           sortFactor = "Sources",
                           sortOrder = sort.order)

## End(Not run)

## Use lmeTest and emmmeans to get predictions and associated statistics

if (requireNamespace("lmerTest", quietly = TRUE) &
    requireNamespace("emmeans", quietly = TRUE))
{
  #Analyse pH
  m1.lmer <- lmerTest::lmer(pH ~ Benches + (Sources * (Type + Species)) +
                              (1|Benches:MainPlots),
                            data=na.omit(tmp))
  TS.emm <- emmeans::emmeans(m1.lmer, specs = ~ Sources:Type)
  TS.preds <- summary(TS.emm)
  den.df <- min(TS.preds$df, na.rm = TRUE)
  ## Modify TS.preds to be compatible with a predictions.frame
  TS.preds <- as.predictions.frame(TS.preds, predictions = "emmean",
                                   se = "SE", interval.type = "CI",
                                   interval.names = c("lower.CL", "upper.CL"))

  ## Form an all.diffs object and check its validity
  TS.vcov <- vcov(TS.emm)
  TS.diffs <- allDifferences(predictions = TS.preds,
                             classify = "Sources:Type",
                             vcov = TS.vcov, tdf = den.df)
  validAlldiffs(TS.diffs)

  #Use sort.alldiffs and save order for use with other response variables
  TS.diffs.sort <- sort(TS.diffs, sortFactor = "Sources",
                        sortParallelToCombo = list(Type = "Control"))
  sort.order <- attr(TS.diffs.sort, which = "sortOrder")

  #Analyse Turbidity
  m2.lmer <- lmerTest::lmer(Turbidity ~ Benches + (Sources * (Type + Species)) +
                              (1|Benches:MainPlots),
                            data=na.omit(tmp))
  TS.emm <- emmeans::emmeans(m2.lmer, specs = ~ Sources:Type)
  TS.preds <- summary(TS.emm)
  den.df <- min(TS.preds$df, na.rm = TRUE)
```

```
  ## Modify TS.preds to be compatible with a predictions.frame
  TS.preds <- as.predictions.frame(TS.preds, predictions = "emmean",
                                   se = "SE", interval.type = "CI",
                                   interval.names = c("lower.CL", "upper.CL"))

  ## Form an all.diffs object, sorting it using the pH sort.order and check its validity
  TS.vcov <- vcov(TS.emm)
  TS.diffs2.sort <- allDifferences(predictions = TS.preds,
                                   classify = "Sources:Type",
                                   vcov = TS.vcov, tdf = den.df,
                                   sortFactor = "Sources",
                                   sortOrder = sort.order)
  validAlldiffs(TS.diffs2.sort)
}
```

`sort.predictions.frame` *Sorts a* `predictions.frame` *according to the predicted values associated with a factor.*

Description

Sorts the rows of a `predictions.frame` according to the predicted values in the `predictions.frame`. These predicted values are generally obtained using `predict.asreml` by specifying a `classify` term comprised of one or more variables. Generally, the values associated with one variable are sorted in parallel within each combination of values of the other variables. When there is more than one variable in the `classify` term, the sorting is controlled using one or more of `sortFactor`, `sortParallelToCombo` and `sortOrder`. If there is only one variable in the `classify` then the `predictions.frame` is sorted according to the order of the complete set of predictions.

Usage

```
## S3 method for class 'predictions.frame'
sort(x, decreasing = FALSE, classify, sortFactor = NULL,
     sortParallelToCombo = NULL, sortNestingFactor = NULL,
     sortOrder = NULL, ...)
```

Arguments

`x`	A `predictions.frame`.
`decreasing`	A `logical` passed to `order` that detemines whether the order is for increasing or decreasing magnitude of the predicted values.
`classify`	A `character` string giving the variables that define the margins of the multiway table that was predicted. Multiway tables are specified by forming an interaction type term from the classifying variables, that is, separating the variable names with the `:` operator.

sortFactor — A character containing the name of the factor that indexes the set of predicted values that determines the sorting of the components. If there is only one variable in the classify term then sortFactor can be NULL and the order is defined by the complete set of predicted values. If there is more than one variable in the classify term then sortFactor must be set. In this case the sortFactor is sorted in the same order within each combination of the values of the sortParallelToCombo variables: the classify variables, excluding the sortFactor. There should be only one predicted value for each unique value of sortFactor within each set defined by a combination of the values of the classify variables, excluding the sortFactor factor. The order to use is determined by either sortParallelToCombo or sortOrder.

sortParallelToCombo — A list that specifies a combination of the values of the factors and numerics, excluding sortFactor, that are in classify. Each of the components of the supplied list is named for a classify variable and specifies a single value for it. The combination of this set of values will be used to define a subset of the predicted values whose order will define the order of sortFactor. Each of the other combinations of the values of the factors and numerics will be sorted in parallel. If sortParallelToCombo is NULL then the first value of each classify variable, except for the sortFactor factor, in the predictions component is used to define sortParallelToCombo. If there is only one variable in the classify then sortParallelToCombo is ignored.

sortNestingFactor — A character containing the name of the factor that defines groups of the sortFactor within which the predicted values are to be ordered. If there is only one variable in the classify then sortNestingFactor is ignored.

sortOrder — A character vector whose length is the same as the number of levels for sortFactor in the predictions.frame. It specifies the desired order of the levels in the reordered the predictions.frame. The argument sortParallelToCombo is ignored.

The following creates a sortOrder vector levs for factor f based on the values in x: levs <- levels(f)[order(x)].

... — further arguments passed to or from other methods. Not used at present.

Details

The basic technique is to change the order of the levels of the sortFactor within the predictions.frame so that they are ordered for a subset of predicted values, one for each levels of the sortFactor. When the classify term consists of more than one variable then a subset of one combination of the values of variables other than the sortFactor, the sortParallelToCombo combination, must be chosen for determining the order of the sortFactor levels. Then the sorting of the rows (and columns) will be in parallel within each combination of the values of sortParallelToCombo variables: the classify term, excluding the sortFactor.

Value

The sorted predictions.frame. Also, the sortFactor and sortOrder attributes are set.

Author(s)

Chris Brien

See Also

as.predictions.frame, print.predictions.frame, sort.alldiffs, predictPlus.asreml, predictPresent.asreml

Examples

```
##Halve WaterRunoff data to reduce time to execute
data(WaterRunoff.dat)
tmp <- subset(WaterRunoff.dat, Date == "05-18")

##Use asreml to get predictions and associated statistics

## Not run:
#Analyse pH
m1.asr <- asreml(fixed = pH ~ Benches + (Sources * (Type + Species)),
                 random = ~ Benches:MainPlots,
                 keep.order=TRUE, data= tmp)
current.asrt <- as.asrtests(m1.asr, NULL, NULL)
current.asrt <- as.asrtests(m1.asr)
current.asrt <- rmboundary(current.asrt)
m1.asr <- current.asrt$asreml.obj

#Get predictions and associated statistics
TS.diffs <- predictPlus.asreml(classify = "Sources:Type",
                               asreml.obj = m1.asr, tables = "none",
                               wald.tab = current.asrt$wald.tab,
                               present = c("Type","Species","Sources"))

#Use sort.predictions.frame and save order for use with other response variables
TS.preds <- TS.diffs$predictions
TS.preds.sort <- sort(TS.preds, sortFactor = "Sources",
                      sortParallelToCombo = list(Type = "Control"))
sort.order <- attr(TS.preds.sort, which = "sortOrder")

#Analyse Turbidity
m2.asr <- asreml(fixed = Turbidity ~ Benches + (Sources * (Type + Species)),
                 random = ~ Benches:MainPlots,
                 keep.order=TRUE, data= tmp)
current.asrt <- as.asrtests(m2.asr)
#Use pH sort.order to sort Turbidity alldiffs object
TS.diffs2 <- predictPlus(m2.asr, classify = "Sources:Type",
                         pairwise = FALSE, error.intervals = "Stand",
                         tables = "none", present = c("Type","Species","Sources"))
TS.preds2 <- TS.diffs2$predictions
TS.preds2.sort <- sort(TS.preds, sortFactor = "Sources", sortOder = sort.order)

## End(Not run)
```

```
## Use lmeTest and emmmeans to get predictions and associated statistics

if (requireNamespace("lmerTest", quietly = TRUE) &
    requireNamespace("emmeans", quietly = TRUE))
{
  #Analyse pH
  m1.lmer <- lmerTest::lmer(pH ~ Benches + (Sources * (Type + Species)) +
                              (1|Benches:MainPlots),
                            data=na.omit(tmp))
  TS.emm <- emmeans::emmeans(m1.lmer, specs = ~ Sources:Type)
  TS.preds <- summary(TS.emm)
  den.df <- min(TS.preds$df, na.rm = TRUE)
  ## Modify TS.preds to be compatible with a predictions.frame
  TS.preds <- as.predictions.frame(TS.preds, predictions = "emmean",
                                   se = "SE", interval.type = "CI",
                                   interval.names = c("lower.CL", "upper.CL"))

  #Use sort.predictions.frame and save order for use with other response variables
  TS.preds.sort <- sort(TS.preds, classify = "Sources:Type", sortFactor = "Sources",
                        sortParallelToCombo = list(Type = "Control"))
  sort.order <- attr(TS.preds.sort, which = "sortOrder")

  #Analyse Turbidity
  m2.lmer <- lmerTest::lmer(Turbidity ~ Benches + (Sources * (Type + Species)) +
                              (1|Benches:MainPlots),
                            data=na.omit(tmp))
  TS.emm <- emmeans::emmeans(m2.lmer, specs = ~ Sources:Type)
  TS.preds <- summary(TS.emm)
  den.df <- min(TS.preds$df, na.rm = TRUE)
  ## Modify TS.preds to be compatible with a predictions.frame
  TS.preds <- as.predictions.frame(TS.preds, predictions = "emmean",
                                   se = "SE", interval.type = "CI",
                                   interval.names = c("lower.CL", "upper.CL"))
}
```

`subset.alldiffs` *Subsets the components in an* `alldiffs.object` *according to the supplied condition.*

Description

Subsets each of the components of an `alldiffs.object`. The subset is determined by applying the condition to the `prediction` component to determine which of its rows are to be included in the subset. Then, if present, this subset is applied to the rows of `backtransforms` and to the rows and columns of `differences`, `p.differences` and `sed` components. In addition, if `sed` is present, `recalcLSD.alldiffs` is called to recalculate the values in the `LSD.frame` stored in the `LSD` component, with any arguments supplied via the `...` argument passed ot it.

The `select` argument of `subset` is not implemented, but can be achieved for variables in the `classify` using the `rmClassifyVars` argument.

Usage

```
## S3 method for class 'alldiffs'
subset(x, subset = rep(TRUE, nrow(x$predictions)),
       rmClassifyVars = NULL, ...)
```

Arguments

x	An alldiffs.object.
subset	A logical that detemines rows of the predictions component of x to be included in the subset. By default all rows are included.
rmClassifyVars	A character that contains the names of the variables in the classify attribute of x that are to be removed from the predictions data.frame and the names of the dimensions of the other components of x. In doing this, the combinations of the remaining classify variables must uniquely index the predictions.
...	further arguments passed to recalcLSD.alldiffs.

Value

An alldiffs.object with the following components of the supplied alldiffs.object subsetted, if present in the original object: predictions, vcov, backtransforms, differences, p.differences and sed. In addition, if sed is present, the LSD.frame in the LSD component will be recalculated.

Author(s)

Chris Brien

See Also

as.alldiffs, allDifferences.data.frame, print.alldiffs, sort.alldiffs, redoErrorIntervals.alldiffs, recalcLSD.alldiffs, predictPlus.asreml, predictPresent.asreml

Examples

```
data(WaterRunoff.dat)

##Use asreml to get predictions and associated statistics

## Not run:
asreml.options(keep.order = TRUE) #required for asreml-R4 only
current.asr <- asreml(fixed = pH ~ Benches + (Sources * (Type + Species)),
                      random = ~ Benches:MainPlots,
                      keep.order=TRUE, data= WaterRunoff.dat)
current.asrt <- as.asrtests(current.asr, NULL, NULL)
TS.diffs <- predictPlus.asreml(classify = "Sources:Type",
                               asreml.obj = current.asr, tables = "none",
                               wald.tab = current.asrt$wald.tab,
                               present = c("Type","Species","Sources"))
```

```
## End(Not run)

## Use lmeTest and emmmeans to get predictions and associated statistics

if (requireNamespace("lmerTest", quietly = TRUE) &
    requireNamespace("emmeans", quietly = TRUE))
{
  m1.lmer <- lmerTest::lmer(pH ~ Benches + (Sources * (Type + Species)) +
                              (1|Benches:MainPlots),
                            data=na.omit(WaterRunoff.dat))
  TS.emm <- emmeans::emmeans(m1.lmer, specs = ~ Sources:Type)
  TS.preds <- summary(TS.emm)
  den.df <- min(TS.preds$df, na.rm = TRUE)
  ## Modify TS.preds to be compatible with a predictions.frame
  TS.preds <- as.predictions.frame(TS.preds, predictions = "emmean",
                                   se = "SE", interval.type = "CI",
                                   interval.names = c("lower.CL", "upper.CL"))

  ## Form an all.diffs object and check its validity
  TS.vcov <- vcov(TS.emm)
  TS.diffs <- allDifferences(predictions = TS.preds, classify = "Sources:Type",
                             vcov = TS.vcov, tdf = den.df)
  validAlldiffs(TS.diffs)
}

## Plot p-values for predictions obtained using asreml or lmerTest
if (exists("TS.diffs"))
{
  ##Use subset.alldiffs to select a subset of the alldiffs object
  TS.diffs.subs <- subset(TS.diffs,
                          subset = grepl("R", Sources, fixed = TRUE) &
                                   Type %in% c("Control","Medicinal"))
}
```

`subset.list` *Forms a* `list` *that contains a subset of the components of the supplied* `list`.

Description

Select components of a `list` specified by a list of numbers or names, or by a logical indicating for each component of the list whether or not it is to be retained.

Usage

```
## S3 method for class 'list'
subset(x, select = 1:length(x), ...)
```

Arguments

x	An list object.
select	A numeric or character that lists or names the components of the list x that are to be retained in the subset. It can also be a logical that is the same length as x and indicates whether or not a component is to be retained.
...	further arguments passed to or from other methods. Not used at present.

Value

A list with just the subset of the components from x. If the components of x are named, then these names are retained in the subset list.

Author(s)

Chris Brien

See Also

subset.alldiffs

Examples

```
x <- list(1:3,letters[1:3],LETTERS[1:3])
y <- subset.list(x, select = c(1,3))
y <- subset.list(x, select = c(TRUE,FALSE,TRUE))

names(x) <- LETTERS[1:3]
y <- subset.list(x, select = c(1,3))
z <- subset.list(x, select = LETTERS[c(1,3)])
x <- list(1:3,letters[1:3],LETTERS[1:3])
names(x)[c(1,3)] <- LETTERS[c(1,3)]
z <- subset.list(x, select = c(1,2))
v <- subset.list(x)
```

---

testranfix.asrtests *Tests for a single fixed or random term in model fitted using* asreml *and records the result in an* asrtests.object.

Description

Tests for a single term, using a REML ratio test (REMLRT) for a random term or based on Wald statistics for a fixed term. The term must be in the fitted model. A random term is removed from the model fit and a REMLRT is performed using REMLRT.asreml. It compares the fit of the model in asreml.obj and the newly fitted model without the term. If the newly fitted model is retained, any boundary terms are then removed using rmboundary.asrtests. For a fixed term, the probability of the Wald statistics is extracted from the pseudo-anova table produced by wald.asreml. If this is available in the asrtests.object, it is used; otherwise wald.asreml is called to add it to the

asrtests.object. Whether nonsignificant terms are dropped is controlled by drop.ran.ns for random terms and drop.fix.ns for fixed terms. A row is added to the test.summary data.frame for the term that is tested.

Usage

```
## S3 method for class 'asrtests'
testranfix(asrtests.obj, term=NULL, alpha = 0.05,
           allow.unconverged = TRUE, allow.fixedcorrelation = TRUE,
           checkboundaryonly = FALSE,
           drop.ran.ns = TRUE, positive.zero = FALSE,
           bound.test.parameters = "none",
           bound.exclusions = c("F","B","S","C"), REMLDF = NULL,
           drop.fix.ns = FALSE, denDF="numeric", dDF.fault = "none",
           dDF.values = NULL, IClikelihood = "none",
           trace = FALSE, update = TRUE,
           set.terms = NULL, ignore.suffices = TRUE,
           bounds = "P", initial.values = NA, ...)
```

Arguments

asrtests.obj	An asrtests.object containing the components (i) asreml.obj, (ii) wald.tab, and (iii) test.summary.
term	A single model term that is valid in asreml, stored as a character. The names of fixed terms must match those in the wald.tab component of the asrtests.obj, while the names of random terms must match those in the vparameters component of the asreml.obj component in the asrtests.obj.
alpha	The significance level for the test.
allow.unconverged	A logical indicating whether to accept a new model even when it does not converge. If FALSE, it will be checked whether convergence can be achieved with the removal of any boundary random terms; random terms will be retested if terms are removed. Also, if FALSE and the fit of the new model has converged, but that of the old model has not, the new model will be accepted.
allow.fixedcorrelation	A logical indicating whether to accept a new model even when it contains correlations in the model whose values have been designated as fixed, bound or singular. If FALSE and the new model contains correlations whose values have not been able to be estimated, the supplied asrtests.obj is returned. The fit in the asreml.obj component of the supplied asrtests.obj will also be tested and a warning issued if both fixed correlations are found in it and allow.fixedcorrelation is FALSE.
checkboundaryonly	If TRUE then boundary and singular terms are not removed by rmboundary.asrtests; a warning is issued instead.
drop.ran.ns	A logical indicating whether to drop a random term from the model when it is nonsignificant. Note that multiple terms specified using a single asreml::at function will only be dropped as a whole. If the term was specified using an

asreml::at function with a single level, then it can be removed and either the level itself or its numeric position in the levels returned by the levels function can be specified in term.

positive.zero Indicates whether the hypothesized values for the variance components being tested are on the boundary of the parameter space. For example, this is true for positively-constrained variance components that, under the reduced model, are zero. This argument does not need to be set if bound.test.parameters is set.

bound.test.parameters
Indicates whether for the variance components being tested, at least some of the hypothesized values are on the boundary of the parameter space. The possibilities are "none", "onlybound" and "one-and-one". The default is "none", although if it is set to "none" and positive.zero is TRUE then bound.test.parameters is taken to be "onlybound". When bound.test.parameters is set to "one-and-one", it signifies that there are two parameters being tested, one of which is bound and the other is not. For example, the latter is true for testing a covariance and a positively-constrained variance component that, under the reduced model, are zero.

bound.exclusions
A character specifying one or more bound (constraint) codes that will result in a variance parameter being excluded from the count of estimated variance parameters in using REMLRT.asreml. If set to NULL then none will be excluded.

REMLDF A numeric giving the difference in the number of variance parameters whose estimates are not of the type specified in bound.exclusions for two models being compared in a REML ratio test using REMLRT.asreml. If NULL then this is determined from the information in the asreml object for the two models.

drop.fix.ns A logical indicating whether to drop a fixed term from the model when it is nonsignificant. Note that multiple terms specified using a single asreml::at function can only be dropped as a whole. If the term was specified using an asreml::at function with a single level, then it can be removed and either the level itself or its numeric position in the levels returned by the levels function can be specified.

denDF Specifies the method to use in computing approximate denominator degrees of freedom when wald.asreml is called. Can be none to suppress the computations, numeric for numerical methods, algebraic for algebraic methods or default, the default, to autommatically choose numeric or algebraic computations depending on problem size. The denominator degrees of freedom are calculated according to Kenward and Roger (1997) for fixed terms in the dense part of the model.

dDF.fault A character specifying the method to use to obtain substitute denominator degrees of freedom. when the numeric or algebraic methods produce faulty values, viz. NA, Inf or less than 0.01. Consistent with when no denDF are available, the default is "residual" and so the residual degrees of freedom from asreml.obj$nedf are used. If dDF.fault = "none", no substitute denominator degrees of freedom are employed; if dDF.fault = "residual"; if dDF.fault = "maximum", the maximum of those denDF that are available, excluding that for the Intercept, is used; if all denDF are faulty, asreml.obj$nedf is used. If dDF.fault = "supplied", a vector of values for the denominator degrees of

freedom is to be supplied in dDF.values. Any other setting is ignored and a warning message produced. Generally, substituting these degrees of freedom is anticonservative in that it is likely that the degrees of freedom used will be too large.

dDF.values — A vector of values to be used when dDF.fault = "supplied". Its values will be used when denDF in a test for a fixed effect is NA. This vector must be the same length as the number of fixed terms, including (Intercept) whose value could be NA.

IClikelihood — A character that controls both the occurence and the type of likelihood for information criterion in the test.summary of the new asrtests.object. If none, none are included. Otherwise, if REML and family is set to asr_guassian (the default), then the AIC and BIC based on the Restricted Maximum Likelihood are included; if full and family is set to asr_guassian, then the AIC and BIC based on the full likelihood, evaluated using REML estimates, are included. If family is asr_binomial or asr_poisson, with dispersion set to 1, the deviance is extracted from object and used to calculate the AIC and BIC. (See also infoCriteria.asreml.)

trace — If TRUE then partial iteration details are displayed when ASReml-R functions are invoked; if FALSE then no output is displayed.

update — If TRUE, and set.terms is NULL, then newfit.asreml is called to fit the model to be tested, using the values of the variance parameters stored in the asreml.object, that is stored in asrtests.obj, as starting values. If FALSE or set.terms is not NULL, then newfit.asreml will not use the stored variance parameter values as starting values when fitting the new model, the only modifications being (i) those for the supplied terms and (ii) those specified via ....

set.terms — A character vector specifying the terms that are to have bounds and/or initial values set prior to fitting. The names must match those in the vparameters component of the asreml.obj component in the new asrtests.object.

ignore.suffices — A logical vector specifying whether the suffices of the asreml-assigned names of the variance terms (i.e. the information to the right of an "!", other than "R!") is to be ignored in matching elements of terms. If TRUE for an element of terms, the suffices are stripped from the asreml-assigned names. If FALSE for an element of terms, the element must exactly match an asreml-assigned name for a variance term. This vector must be of length one or the same length as terms. If it is of length one then the same action is applied to the asreml-assigned suffices for all the terms in terms.

bounds — A character vector specifying the bounds to be applied to the terms specified in set.terms. This vector must be of length one or the same length as set.terms. If it is of length one then the same constraint is applied to all the terms in set.terms. If any of the bounds are equal to NA then they are left unchanged for those terms.

initial.values — A character vector specifying the initial values for the terms specified in terms. This vector must be of length one or the same length as terms. If it is of length one then the same initial value is applied to all the terms in terms. If any of the initial.values are equal to NA then they are left unchanged for those terms.

... — Further arguments passed to asreml, wald.asreml and as.asrtests.

Value

An asrtests.object containing the components (i) asreml.obj, (ii) wald.tab, and (iii) test.summary. If the term is not in the model, then the supplied asreml object will be returned. Also, reml.test will have the likelihood ratio and the p-value set to NA and the degrees of freedom to zero. Similarly, the row of test.summary for the term will have its name, DF set to NA, p-value set to NA, and action set to Absent.

Author(s)

Chris Brien

References

Kenward, M. G., & Roger, J. H. (1997). Small sample inference for fixed effects from restricted maximum likelihood. *Biometrics*, **53**, 983-997.

See Also

asremlPlus-package, as.asrtests, chooseModel.asrtests, REMLRT.asreml, rmboundary.asrtests, newfit.asreml, changeModelOnIC.asrtests, changeTerms.asrtests, reparamSigDevn.asrtests

Examples

```
## Not run:
data(Wheat.dat)
current.asr <- asreml(yield ~ Rep + WithinColPairs + Variety,
                      random = ~ Row + Column + units,
                      residual = ~ ar1(Row):ar1(Column),
                      data=Wheat.dat)
current.asrt <- as.asrtests(current.asr, NULL, NULL)
current.asrt <- rmboundary(current.asrt)
# Test nugget term
current.asrt <- testranfix(current.asrt, "units", positive=TRUE)

## End(Not run)
```

---

testresidual.asrtests *Fits a new residual formula, tests whether the change is significant and records the result in an asrtests.object.*

Description

Fits a new residual formula using asreml-R4 (replaces the rcov formula of asreml-R3) and tests whether the change is significant. If simpler = FALSE the model to be fitted must be more complex than the one whose fit has been stored in asrtests.obj. That is, the new model must have more parameters. However, if simpler = TRUE the model to be fitted must be simpler than the one whose fit has been stored in asrtests.obj in that it must have fewer parameters. Any boundary terms are

removed using rmboundary.asrtests, which may mean that the models are not nested. The test is a REML likelihood ratio test that is performed using REMLRT.asreml, which is only valid if the models are nested. It compares the newly fitted model with the fit of the model in asrtest.obj. If the two models have the same number of variance parameters, then no change is made to the residual. A row is added to the test.summary data.frame using the supplied label.

Usage

```
## S3 method for class 'asrtests'
testresidual(asrtests.obj, terms=NULL, label = "R model",
             simpler = FALSE, alpha = 0.05,
             allow.unconverged = TRUE, allow.fixedcorrelation = TRUE,
             checkboundaryonly = FALSE, positive.zero = FALSE,
             bound.test.parameters = "none",
             bound.exclusions = c("F","B","S","C"), REMLDF = NULL,
             denDF="numeric", IClikelihood = "none",
             update = TRUE, trace = FALSE,
             set.terms = NULL, ignore.suffices = TRUE,
             bounds = "P", initial.values = NA, ...)
```

Arguments

asrtests.obj	an asrtests.object for a fitted model that is a list containing the componets (i) asreml.obj, (ii) wald.tab (iii) test.summary.
terms	A model for the residual argument in asreml-R4 (the rcov formula in older versions of asreml), stored as a character. To remove the model, enter "-(.)".
label	A character string to use as the label in test.summary and which indicates what is being tested.
simpler	A logical indicating whether the new model to be fitted is simpler than the already fitted model whose fit is stored in asrtests.obj.
alpha	The significance level for the test.
allow.unconverged	A logical indicating whether to accept a new model even when it does not converge. If FALSE and the fit of the new model does not converge, the supplied asreml object is returned. Also, if FALSE and the fit of the new model has converged, but that of the old model has not, the new model will be accepted.
allow.fixedcorrelation	A logical indicating whether to accept a new model even when it contains correlations in the model whose values have been designated as fixed, bound or singular. If FALSE and the new model contains correlations whose values have not been able to be estimated, the supplied asrtests.obj is returned. The fit in the asreml.obj component of the supplied asrtests.obj will also be tested and a warning issued if both fixed correlations are found in it and allow.fixedcorrelation is FALSE.
checkboundaryonly	If TRUE then boundary and singular terms are not removed by rmboundary.asrtests; a warning is issued instead.

positive.zero Indicates whether the hypothesized values for the variance components being tested are on the boundary of the parameter space. For example, this is true for positively-constrained variance components that, under the reduced model, are zero. This argument does not need to be set if bound.test.parameters is set.

bound.test.parameters
Indicates whether for the variance components being tested, at least some of the hypothesized values are on the boundary of the parameter space. The possibilities are "none", "onlybound" and "one-and-one". The default is "none", although if it is set to "none" and positive.zero is TRUE then bound.test.parameters is taken to be "onlybound". When bound.test.parameters is set to "one-and-one", it signifies that there are two parameters being tested, one of which is bound and the other is not. For example, the latter is true for testing a covariance and a positively-constrained variance component that, under the reduced model, are zero.

bound.exclusions
A character specifying one or more bound (constraint) codes that will result in a variance parameter being excluded from the count of estimated variance parameters in using REMLRT.asreml. If set to NULL then none will be excluded.

REMLDF A numeric giving the difference in the number of variance parameters whose estimates are not of the type specified in bound.exclusions for two models being compared in a REML ratio test using REMLRT.asreml. If NULL then this is determined from the information in the asreml object for the two models.

denDF Specifies the method to use in computing approximate denominator degrees of freedom when wald.asreml is called. Can be none to suppress the computations, numeric for numerical methods, algebraic for algebraic methods or default, the default, to autommatically choose numeric or algebraic computations depending on problem size. The denominator degrees of freedom are calculated according to Kenward and Roger (1997) for fixed terms in the dense part of the model.

IClikelihood A character that controls both the occurence and the type of likelihood for information criterion in the test.summary of the new asrtests.object. If none, none are included. Otherwise, if REML, then the AIC and BIC based on the Restricted Maximum Likelihood are included; if full, then the AIC and BIC based on the full likelihood, evaluated using REML estimates, are included. (See also infoCriteria.asreml.)

update If TRUE, and set.terms is NULL, then newfit.asreml is called to fit the model to be tested, using the values of the variance parameters stored in the asreml.object, that is stored in asrtests.obj, as starting values. If FALSE or set.terms is not NULL, then newfit.asreml will not use the stored variance parameter values as starting values when fitting the new model, the only modifications being (i) the residual (rcov) model is that specified in terms (ii) those specified via ....

trace If TRUE then partial iteration details are displayed when ASReml-R functions are invoked; if FALSE then no output is displayed.

set.terms A character vector specifying the terms that are to have bounds and/or initial values set prior to fitting. The names must match those in the vparameters component of the asreml.obj component in the new asrtests.object.

ignore.suffices
: A logical vector specifying whether the suffices of the asreml-assigned names of the variance terms (i.e. the information to the right of an "!", other than "R!") is to be ignored in matching elements of terms. If TRUE for an element of terms, the suffices are stripped from the asreml-assigned names. If FALSE for an element of terms, the element must exactly match an asreml-assigned name for a variance term. This vector must be of length one or the same length as terms. If it is of length one then the same action is applied to the asreml-assigned suffices for all the terms in terms.

bounds
: A character vector specifying the bounds to be applied to the terms specified in set.terms. This vector must be of length one or the same length as set.terms. If it is of length one then the same constraint is applied to all the terms in set.terms. If any of the bounds are equal to NA then they are left unchanged for those terms.

initial.values
: A character vector specifying the initial values for the terms specified in terms. This vector must be of length one or the same length as terms. If it is of length one then the same initial value is applied to all the terms in terms. If any of the initial.values are equal to NA then they are left unchanged for those terms.

...
: Further arguments passed to asreml, wald.asreml and as.asrtests.

Value

An asrtests.object containing the components (i) asreml.obj, (ii) wald.tab, and (iii) test.summary. If the term is not in the model, then the supplied asreml.obj will be returned. Also, reml.test will have the likelihood ratio and the p-value set to NA and the degrees of freedom to zero. Similarly, the row of test.summary for the term will have its name, a p-value set to NA, and action set to Absent.

Author(s)

Chris Brien

References

Kenward, M. G., & Roger, J. H. (1997). Small sample inference for fixed effects from restricted maximum likelihood. *Biometrics*, **53**, 983-997.

See Also

asremlPlus-package, as.asrtests, changeTerms.asrtests, chooseModel.asrtests, REMLRT.asreml, rmboundary.asrtests, newfit.asreml, testswapran.asrtests, changeModelOnIC.asrtests, changeTerms.asrtests, reparamSigDevn.asrtests

Examples

```
## Not run:
data(Wheat.dat)
current.asr <- asreml(yield ~ Rep + WithinColPairs + Variety,
```

```
                    random = ~ Row + Column + units,
                    residual = ~ ar1(Row):ar1(Column),
                    data=Wheat.dat)
current.asrt <- as.asrtests(current.asr, NULL, NULL)
current.asrt <- rmboundary(current.asrt)
# Test Row autocorrelation
current.asrt <- testresidual(current.asrt, "~ Row:ar1(Column)",
                             label="Row autocorrelation", simpler=TRUE)
print(current.asrt)

## End(Not run)
```

`testswapran.asrtests` *Tests, using a REMLRT, the significance of the difference between the current random model and one in which oldterms are dropped and newterms are added. The result is recorded in an* `asrtests.object`.

Description

Fits a new random model using `asreml` by removing `oldterms` and adding `newterms`. If `simpler = FALSE` the model to be fitted must be more complex than the one whose fit has been stored in `asrtests.obj`. That is, the new model must have more parameters. However, if `simpler = TRUE` the model to be fitted must be simpler than the one whose fit has been stored in `asrtests.obj` in that it must have fewer parameters. The test is a REML ratio test that is performed using `REMLRT.asreml`, **which is only valid if the models are nested**. It compares the newly fitted model with the fit of the model in `asrtests.obj`. A row is added to the `test.summary data.frame` using the supplied `label`. If the newly fitted model is retained, any boundary terms are then removed using `rmboundary.asrtests`. If the models are not nested, then using `changeModelOnIC.asrtests` may be the more appropriate approach for comparing models.

Usage

```
## S3 method for class 'asrtests'
testswapran(asrtests.obj, oldterms = NULL, newterms = NULL,
            label = "Swap in random model", simpler = FALSE, alpha = 0.05,
            allow.unconverged = TRUE, allow.fixedcorrelation = TRUE,
            checkboundaryonly = FALSE,
            positive.zero = FALSE, bound.test.parameters = "none",
            bound.exclusions = c("F","B","S","C"), REMLDF = NULL,
            denDF="numeric", IClikelihood = "none",
            trace = FALSE, update = TRUE,
            set.terms = NULL, ignore.suffices = TRUE,
            bounds = "P", initial.values = NA, ...)
```

Arguments

`asrtests.obj`	an `asrtests.object` for a fitted model that is a list containing the components (i) `asreml.obj`, (ii) `wald.tab` (iii) `test.summary`.

oldterms Terms, stored as a character, that are to be removed from the random model using asreml. The names of the terms must match those in the vparameters component of the asreml.obj component in asrtests.obj. Note that multiple terms specified using a single asreml::at function can only be dropped as a whole. If the term was specified using an asreml::at function with a single level, then it can be removed and either the level itself or its numeric position in the levels returned by the levels function can be specified.

newterms Terms, stored as a character, that are to be added to the random model using asreml.

simpler A logical indicating whether the new model to be fitted. after the changes made as a result of swapping oldterms for newterms, is simpler than the already fitted model whose fit is stored in asrtests.obj.

alpha The significance level for the test.

allow.unconverged
A logical indicating whether to accept a new model even when it does not converge. If FALSE and the fit of the new model does not converge, the supplied asrtests.obj is returned. Also, if FALSE and the fit of the new model has converged, but that of the old model has not, the new model will be accepted.

allow.fixedcorrelation
A logical indicating whether to accept a new model even when it contains correlations in the model whose values have been designated as fixed, bound or singular. If FALSE and the new model contains correlations whose values have not been able to be estimated, the supplied asrtests.obj is returned. The fit in the asreml.obj component of the supplied asrtests.obj will also be tested and a warning issued if both fixed correlations are found in it and allow.fixedcorrelation is FALSE.

checkboundaryonly
If TRUE then boundary and singular terms are not removed by rmboundary.asrtests; a warning is issued instead.

label A character string to use as the label in test.summary and which indicates what is being tested.

positive.zero Indicates whether the hypothesized values for the variance components being tested are on the boundary of the parameter space. For example, this is true for positively-constrained variance components that, under the reduced model, are zero. This argument does not need to be set if bound.test.parameters is set.

bound.test.parameters
Indicates whether for the variance components being tested, at least some of the hypothesized values are on the boundary of the parameter space. The possibilities are "none", "onlybound" and "one-and-one". The default is "none", although if it is set to "none" and positive.zero is TRUE then bound.test.parameters is taken to be "onlybound". When bound.test.parameters is set to "one-and-one", it signifies that there are two parameters being tested, one of which is bound and the other is not. For example, the latter is true for testing a covariance and a positively-constrained variance component that, under the reduced model, are zero.

bound.exclusions
: A character specifying one or more bound (constraint) codes that will result in a variance parameter being excluded from the count of estimated variance parameters in using REMLRT.asreml. If set to NULL then none will be excluded.

REMLDF
: A numeric giving the difference in the number of variance parameters whose estimates are not of the type specified in bound.exclusions for two models being compared in a REML ratio test using REMLRT.asreml. If NULL then this is determined from the information in the asreml object for the two models.

denDF
: Specifies the method to use in computing approximate denominator degrees of freedom when wald.asreml is called. Can be none to suppress the computations, numeric for numerical methods, algebraic for algebraic methods or default, the default, to automatically choose numeric or algebraic computations depending on problem size. The denominator degrees of freedom are calculated according to Kenward and Roger (1997) for fixed terms in the dense part of the model.

IClikelihood
: A character that controls both the occurrence and the type of likelihood for information criterion in the test.summary of the new asrtests.object. If none, none are included. Otherwise, if REML, then the AIC and BIC based on the Restricted Maximum Likelihood are included; if full, then the AIC and BIC based on the full likelihood, evaluated using REML estimates, are included. (See also infoCriteria.asreml.)

trace
: If TRUE then partial iteration details are displayed when ASReml-R functions are invoked; if FALSE then no output is displayed.

update
: If TRUE, and set.terms is NULL, then newfit.asreml is called to fit the model to be tested, using the values of the variance parameters stored in the asreml.object, that is stored in asrtests.obj, as starting values. If FALSE or set.terms is not NULL, then newfit.asreml will not use the stored variance parameter values as starting values when fitting the new model, the only modifications being (i) for the supplied oldterms and (ii) those specified via ....

set.terms
: A character vector specifying the terms that are to have bounds and/or initial values set prior to fitting. The names must match those in the vparameters component of the asreml.obj component in the asrtests.object.

ignore.suffices
: A logical vector specifying whether the suffices of the asreml-assigned names of the variance terms (i.e. the information to the right of an "!", other than "R!") is to be ignored in matching elements of terms. If TRUE for an element of terms, the suffices are stripped from the asreml-assigned names. If FALSE for an element of terms, the element must exactly match an asreml-assigned name for a variance term. This vector must be of length one or the same length as terms. If it is of length one then the same action is applied to the asreml-assigned suffices for all the terms in terms.

bounds
: A character vector specifying the bounds to be applied to the terms specified in set.terms. This vector must be of length one or the same length as set.terms. If it is of length one then the same constraint is applied to all the terms in set.terms. If any of the bounds are equal to NA then they are left unchanged for those terms.

initial.values	A character vector specifying the initial values for the terms specified in terms. This vector must be of length one or the same length as terms. If it is of length one then the same initial value is applied to all the terms in terms. If any of the initial.values are equal to NA then they are left unchanged for those terms.
...	Further arguments passed to asreml, wald.asreml and as.asrtests.

Value

An asrtests.object for a fitted model that is a list containing the components (i) asreml.obj, (ii) wald.tab (iii) test.summary. If the term is not in the model, then the supplied asreml object will be returned. Also, reml.test will have the likelihood ratio and the p-value set to NA and the degrees of freedom to zero. Similarly, the row of test.summary for the term will have its name, a p-value set to NA, and action set to Absent.

Author(s)

Chris Brien

References

Kenward, M. G., & Roger, J. H. (1997). Small sample inference for fixed effects from restricted maximum likelihood. *Biometrics*, **53**, 983-997.

See Also

as.asrtests, chooseModel.asrtests, REMLRT.asreml, rmboundary.asrtests, newfit.asreml, testresidual.asrtests, changeModelOnIC.asrtests, changeTerms.asrtests, reparamSigDevn.asrtests

Examples

```
## Not run:
current.asrt <- testswapran(current.asrt, oldterms = "str(~ Cart/xDays, ~us(2):id(184))",
                            newterms = "Cart/xDays", pos = FALSE,
                            label = "Intercept/Slope correlation",
                            simpler = TRUE)
  print(current.asrt)

## End(Not run)
```

---

validAlldiffs *Checks that an object is a valid alldiffs object.*

Description

Checks that an object is an alldiffs.object of S3-class alldiffs containing the components asreml.obj, wald.tab and test.summary.

Usage

```
validAlldiffs(object)
```

Arguments

object	an alldiffs.object.

Value

TRUE or a character describing why the object is not a valid alldiffs.object.

Author(s)

Chris Brien

See Also

alldiffs.object, is.alldiffs, as.alldiffs,
validPredictionsFrame, validAsrtests

Examples

```
data(Oats.dat)

## Use lmerTest and emmmeans to get predictions and associated statistics
if (requireNamespace("lmerTest", quietly = TRUE) &
    requireNamespace("emmeans", quietly = TRUE))
{
  m1.lmer <- lmerTest::lmer(Yield ~ Nitrogen*Variety + (1|Blocks/Wplots),
                            data=Oats.dat)
  Var.emm <- emmeans::emmeans(m1.lmer, specs = ~ Nitrogen:Variety)
  Var.preds <- summary(Var.emm)
  den.df <- min(Var.preds$df)
  ## Modify Var.preds to be compatible with a predictions.frame
  Var.preds <- as.predictions.frame(Var.preds, predictions = "emmean",
                                    se = "SE", interval.type = "CI",
                                    interval.names = c("lower.CL", "upper.CL"))
  Var.vcov <- vcov(Var.emm)
  Var.sed <- NULL

  ## Form an all.diffs object
   Var.diffs <- as.alldiffs(predictions = Var.preds, classify = "Nitrogen:Variety",
                            sed = Var.sed, vcov = Var.vcov, tdf = den.df)

  ## check the validity of Var.diffs
  validAlldiffs(Var.diffs)
}
```

validAsrtests *Checks that an object is a valid asrtests object.*

Description

Checks that an object is an asrtests.object of S3-class asrtests containing the components asreml.obj, wald.tab and test.summary.

Usage

```
validAsrtests(object)
```

Arguments

object	an asrtests.object.

Value

TRUE or a character describing why the object is not a valid asrtests.object.

Author(s)

Chris Brien

See Also

asrtests.object, is.asrtests, as.asrtests,
validPredictionsFrame, validAlldiffs

Examples

```
## Not run:
library(dae)
library(asreml)
library(asremlPlus)
## use ?Wheat.dat for data set details
data(Wheat.dat)

# Fit initial model
current.asr <- asreml(yield ~ Rep + WithinColPairs + Variety,
                      random = ~ Row + Column + units,
                      residual = ~ ar1(Row):ar1(Column),
                      data=Wheat.dat)

# Load current fit into an asrtests object
current.asrt <- as.asrtests(current.asr, NULL, NULL)

# check validity of current.asrt
validAsrtests(current.asrt)
```

```
## End(Not run)
```

---

validPredictionsFrame *Checks that an object is a valid predictions.frame.*

Description

Checks that an object is a predictions.frame of S3-class data.frame that contains the columns predicted.value, standard.error and est.status.

Usage

```
validPredictionsFrame(object)
```

Arguments

object	an predictions.frame.

Value

TRUE or a character describing why the object is not a valid predictions.frame.

Author(s)

Chris Brien

See Also

predictions.frame, is.predictions.frame, as.predictions.frame, validAsrtests, validAlldiffs

Examples

```
data(Oats.dat)

## Use asreml to get predictions and associated statistics

## Not run:
m1.asr <- asreml(Yield ~ Nitrogen*Variety,
                 random=~Blocks/Wplots,
                 data=Oats.dat)
current.asrt <- as.asrtests(m1.asr)
Var.pred <- asreml::predict.asreml(m1.asr, classify="Nitrogen:Variety",
                                   sed=TRUE)
if (getASRemlVersionLoaded(nchar = 1) == "3")
  Var.pred <- Var.pred$predictions
Var.preds <- as.predictions.frame(Var.pred$pvals, se = "std.error",
                                  est.status = "status")
```

```
## End(Not run)

  ## Use lmerTest and emmmeans to get predictions and associated statistics
  if (requireNamespace("lmerTest", quietly = TRUE) &
      requireNamespace("emmeans", quietly = TRUE))
  {
    m1.lmer <- lmerTest::lmer(Yield ~ Nitrogen*Variety + (1|Blocks/Wplots),
                              data=Oats.dat)
    Var.emm <- emmeans::emmeans(m1.lmer, specs = ~ Nitrogen:Variety)
    Var.preds <- summary(Var.emm)
    Var.preds <- as.predictions.frame(Var.preds, predictions = "emmean",
                                      se = "SE", interval.type = "CI",
                                      interval.names = c("lower.CL", "upper.CL"))
  }

  if (exists("Var.preds"))
  {
    ## Check the class and validity of the predictions.frame
    is.predictions.frame(Var.preds)
    validPredictionsFrame(Var.preds)
  }
```

`variofaces.asreml` *Plots empirical variogram faces, including envelopes, as described by Stefanova, Smith & Cullis (2009).*

Description

A function that produces a plot for each face of an empirical 2D `variogram` based on residuals produced after the fitting of a model using the function `asreml`. It also adds envelopes to the plot by simulating data sets in parallel from a multivariate normal distribution with expectation equal to the fitted values obtained from the fixed and spline terms and variance matrix equal to the fitted variance matrix (Stefanova, Smith & Cullis, 2009). The plot is controlled by the `residual` model, which must consist of two factors corresponding to the two physical dimensions underlying the data. It can also have a third term involving the `at` or `dsum` function that defines sections of the data, such as experiments in different environments. In this case, the two variogram faces are produced for each section.

Usage

```
## S3 method for class 'asreml'
variofaces(asreml.obj, means=NULL, V=NULL,
           sections = NULL, row.factor = NULL, col.factor = NULL,
           nsim=100, seed = NULL,
           extra.matrix = NULL, ignore.terms = NULL, fixed.spline.terms = NULL,
           bound.exclusions = c("F","B","S","C"), tolerance=1E-10,
           units = "ignore", update = TRUE, trace = FALSE,
           graphics.device=NULL, ncores = 2, ...)
```

Arguments

asreml.obj	An asreml object from a call to asreml in which the data argument has been set.
means	The vector of means to be used in generating simulated data sets. If it is NULL, the fitted values based on object are used. It must be the same length as the response variable for object.
V	The fitted variance matrix, i.e. having the appropriate pattern and values given the model fitted to the observed data and the estimates of the parameters obtained. If V is NULL then estimateV.asreml is called to obtain it from asreml.obj
sections	A single character string that specifies the name of the column in the data.frame that contains the factor that identifies different sections of the data to which separate spatial models have been fitted.
row.factor	A single character string nominating a factor that indexes the rows of a grid that are one dimension of a spatial correlation model. The factor must a column in the data.frame stored in the asreml.obj.
col.factor	A single character string nominating a factor that indexes the columns of a grid that are one dimension of a spatial correlation model. The factor must a column in the data.frame stored in the asreml.obj.
nsim	The number of data sets to be simulated in obtaining the envelopes.
seed	A single value, interpreted as an integer, that specifies the starting value of the random number generator. The "L'Ecuyer-CMRG" random generator is used and nextRNGStream is used to seed each core from the original seed.
extra.matrix	A matrix of order equal to the number of observations that is to be added to the variance matrix, the latter based on the information in asreml.obj. It is assumed that the sigma-parameterized values of the variance parameter estimates, such as is given in the varcomp component of summary.asreml, have been used in calculating extra.matrix; the values in the vparameters component of G.param and R.param may be either gamma- or sigma-parameterized. The argument extra.matrix can be used in conjunction with ignore.terms as a workaround to include components of the variance matrix for variance functions that have not been implemented in estimateV.
ignore.terms	A character giving terms from either the random or residual models that are to be ignored in that their contributions to the variance is not to be included in the estimated matrix. The term names are those given in the vparameters component of the asreml object or the varcomp component produced by summary.asreml, but only up to the first exclamation mark (!). This can be used in conjunction with estimateV.asreml as a workaround to include components of the variance matrix for variance functions that have not been implemented in estimateV.
fixed.spline.terms	A character vector giving one or more spline terms in the random model that are regarded as fixed and so are to be ignored because they are not regarded as contributing to the variance. The term names are those given in the vparameters component of the asreml object or the varcomp component produced by summary.asreml, but only up to the first exclamation mark (!).

bound.exclusions
: A character specifying one or more bound codes that will result in a variance parameter in the random model being excluded from contributing to the variance. If set to NULL then none will be excluded.

tolerance
: The value such that eigenvalues less than it are considered to be zero.

units
: A character indicating whether the BLUPs for units are added to the residuals when this reserved factor is included in the random model. Possible values are addtoresiduals and ignore. If standardized conditional residuals are plotted and the BLUPs for units are to be added then it is the standardized BLUPs that are added.

update
: If TRUE, and set.terms is NULL, then newfit.asreml is called to fit the model to be tested, using the values of the variance parameters stored in the asreml.object, that is stored in asrtests.obj, as starting values. If FALSE or set.terms is not NULL, then newfit.asreml will not use the stored variance parameter values as starting values when fitting the new model, the only modifications being ((i) the model is fitted to simulated data and (ii) those specified via ..., except that changes cannot be made to any of the models.

trace
: If TRUE then partial iteration details are displayed when ASReml-R functions are invoked; if FALSE then no output is displayed.

graphics.device
: A character specifying a graphics device for plotting. The default is graphics.device = NULL, which will result in plots being produced on the current graphics device. Setting it to "windows", for example, will result in a windows graphics device being opened.

ncores
: A numeric specifying the number of cores to use in doing the simulations. In choosing a value for ncores, it is necessary to take into account other processes that are using parallel processing at the same time.

...
: Other arguments that are passed down to the function asreml. Changes to the models are not allowed. Other changes are dangerous and generally should be avoided.

Details

The residual model is scanned to ensure that it involves only two factors not included in the at function, and to see if it has a third factor in an at function. If so, the faces of the 2D variogram, each based on one of the two non-at factors, are derived from the residuals in the supplied asreml object using asreml.variogram, this yielding the observed variogram faces. If aom was set to TRUE for the asreml object, the standardized conditional residuals are used. Then nsim data sets are generated by adding the fitted.values, extracted from the asreml object, to a vector of values randomly generated from a normal distribution with expectation zero and variance matrix V. Each data set is analyzed using the model in object and several sets are generated and analyzed in parallel. The variogram values for the faces are obtained using asreml.variogram stored. Note, if the analysis for a data set does not converge in maxiter iterations, it is discarded and a replacement data set generated. The value of maxiter can be specified in the call to variofaces.asreml. Plots are produced for each face and include the observed values and the 2.5%, 50% & 97.5% quantiles.

Value

A `list` with the following components:

1. **face1:** a `data.frame` containing the variogram values on which the plot for the first dimension is based.
2. **face2:** a `data.frame` containing the variogram values on which the plot for the second dimension is based.

Author(s)

Chris Brien

References

Stefanova, K. T., Smith, A. B. & Cullis, B. R. (2009) Enhanced diagnostics for the spatial analysis of field trials. *Journal of Agricultural, Biological, and Environmental Statistics*, **14**, 392–410.

See Also

asremlPlus-package, asreml, newfit.asreml, plotVariofaces.data.frame, simulate.asreml, set.seed.

Examples

```
## Not run:
data(Wheat.dat)
current.asr <- asreml(yield ~ Rep + WithinColPairs + Variety,
                      random = ~ Row + Column + units,
                      residual = ~ ar1(Row):ar1(Column),
                      data=Wheat.dat)
current.asrt <- as.asrtests(current.asr, NULL, NULL)
current.asrt <- rmboundary.asrtests(current.asrt)
# Form variance matrix based on estimated variance parameters
s2 <- current.asr$sigma2
gamma.Row <- current.asr$gammas[1]
gamma.unit <- current.asr$gammas[2]
rho.r <- current.asr$gammas[4]
rho.c <- current.asr$gammas[5]
row.ar1 <- mat.ar1(order=10, rho=rho.r)
col.ar1 <- mat.ar1(order=15, rho=rho.c)
V <- gamma.Row * fac.sumop(Wheat.dat$Row) +
     gamma.unit * diag(1, nrow=150, ncol=150) +
     mat.dirprod(col.ar1, row.ar1)
V <- s2*V

#Produce variogram faces plot (Stefanaova et al, 2009)
variofaces(current.asr, V=V, ncores = parallel::detectCores())

## End(Not run)
```

WaterRunoff.dat — *Data for an experiment to investigate the quality of water runoff over time*

Description

This data is from an experiment to investigate the quality of water runoff. However, it has been modified to hide the true identity of the Species and Sources. It is used to provide executable examples of the functions listed under **Examples**.

Usage

```
data(WaterRunoff.dat)
```

Format

A data.frame containing 440 observations of 13 variables.

Author(s)

Chris Brien

Source

Kazemi, F. (pers. comm.)

See Also

`chooseModel.asrtests`, `reparamSigDevn.asrtests`, `plotPredictions.data.frame`, `predictPlus.asreml`, `predictPresent.asreml`

Wheat.dat — *Data for a 1976 experiment to investigate 25 varieties of wheat*

Description

The data appears in Gilmour et al. (1995) and is from a field experiment designed to compare the performance of 25 varieties of spring wheat. An analysis of it using asreml is presented by Butler et al. (2023, Section 7.6), although they suggest that it is a barley experiment. It is used in the Wheat vignettes [Enter `vignette(package = "asremlPlus")`] as an executable example of the use of the `asremlPlus` to analyse a data set.

The experiment was conducted at Slate Hall Farm, UK, in 1976 and was designed as a balanced lattice square with 6 replicates laid out in a 10×15 rectangular grid. The columns in the data frame are: Rep, Row, Column, WithinColPairs, Variety, yield. The response variable is the grain yield.

Usage

```
data(Wheat.dat)
```

Format

A data.frame containing 150 observations of 6 variables.

Author(s)

Chris Brien

Source

Butler, D. G., Cullis, B. R., Gilmour, A. R., Gogel, B. J. and Thompson, R. (2023). *ASReml-R Reference Manual Version 4.2.* VSN International Ltd, https://asreml.kb.vsni.co.uk/.

Gilmour, A. R., et al. (1995) Average Information REML: An efficient algorithm for variance parameter estimation in linear mixed models. *Biometrics*, **51**, 1440-1450.

Index